TURING 图灵程序设计丛书

PACKT PUBLISHING

Mastering Machine Learning with R Second Edition

精通机器学习：基于R（第2版）

【美】Cory Leismester 著
陈光欣 译

- 利用R包轻松应用机器学习方法
- 展示各类机器学习方法的优势与潜在问题
- 技术与理论并重，通过丰富的商业案例实现机器学习高级概念

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

数字版权声明

作者简介

Cory Leismester

具有十多年量化管理经验，目前在银行业担任高级量化管理经理，负责构建市场营销和监管模型。曾在礼来公司就职16年，负责销售、市场调查、精益六西格玛、营销分析、新产品预测等工作。

封面设计 | 83720326@qq.com
自在书装设计

TURING 图灵程序设计丛书

Mastering Machine Learning with R Second Edition

精通机器学习：基于R

（第2版）

【美】Cory Leismester 著
陈光欣 译

人民邮电出版社
北京

图书在版编目（C I P）数据

精通机器学习 : 基于R / (美) 考瑞·莱斯米斯特尔
著 ; 陈光欣 译. -- 2版. -- 北京 : 人民邮电出版社,
2018.2
（图灵程序设计丛书）
ISBN 978-7-115-47778-1

Ⅰ. ①精… Ⅱ. ①考… ②陈… Ⅲ. ①机器学习②程
序语言—程序设计 Ⅳ. ①TP181②TP312

中国版本图书馆CIP数据核字(2018)第010916号

版 权 声 明

内 容 提 要

机器学习是近年来的热门技术话题，R 语言是处理其中大量数据的有力工具。本书为读者提供机器学习和 R 语言的坚实算法基础和业务基础，内容包括机器学习基本概念、线性回归、逻辑斯蒂回归和判别分析、线性模型的高级特征选择、K 最近邻和支持向量机等，力图平衡实践中的技术和理论两方面。

本书适合想理解和表述机器学习算法的 IT 人士、想在分析中发挥 R 强大威力的统计学专家。即使是同时精通 IT 技术和统计学的读者，在本书中仍然可以发现一些有用的窍门和技巧。

◆ 著　　[美] Cory Lesmeister
译　　陈光欣
责任编辑　陈　曦
责任印制　周昇亮

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　　　　印刷

◆ 开本：800×1000　1/16
印张：19.5
字数：461千字　　　　2018年 2 月第 1 版
印数：1 – 4 000册　　　　2018年 2 月北京第 1 次印刷

著作权合同登记号　图字：01-2017-5046号

定价：69.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广登字 20170147 号

前　言

“应该给人第二次机会，但一定要留个心眼儿。”

——约翰 · 韦恩

人生中，能得到第二次机会可不常见。我还记得完成本书第1版的编辑工作之后，我不停地问自己：“为什么不……？”或者“我都写了些什么东西啊？”实际上，本书第1版出版之后，我做的第一个项目没有使用书中的任何一种方法。我暗下决心，如果还有机会，一定要在第2版中介绍这些方法。

当我开始写作第1版时，目标是做出点不一样的东西，在介绍各种机器学习方法的同时，还要使内容喜闻乐见。收到所有反馈之后，我认为自己实现了这个目标。但事物总是不完美的，而且，如果你想满足所有人的需要，那最终谁都满足不了。我想起了自己最喜欢的那句腓特烈大帝的名言：“诸事皆殚精竭虑者，终将一无所成。”所以，我并非一味求全，而是提供足够的技能和工具，来使读者尽量轻松愉快地学习R语言和机器学习。在第1版的基础之上，我又添加了一些非常有趣的新技术。总会有一些批评者抱怨这本书没有提供足够的数学知识，或是缺少某些方面的内容。我对这些意见的回答是：它们已经存在！为什么因为有人抱怨就要重复那些已经有人做了，并且做得非常好的事情呢？再次声明，我要写出一些与众不同的东西，一些能够抓住读者眼球并能使他们在这个充满竞争的领域取得成功的东西。

给出第2版每章内容的修改（或改进）之前，我先解释一下第2版总体上的变化。第一个总体变化就是，我放弃了一直使用`=`作为赋值操作符（而不是使用`<-`）的努力。当我越来越多地与他人分享代码时，我意识到再也不能使用`=`，而应该使用`<-`了。签下第2版合约之后，我做的第一件事就是逐行检查代码，将`=`修改为`<-`。第2版更重要的一个改变是，代码更加整洁和标准化，这对于与合作者和管理者（恕我直言）分享代码也非常重要。使用版本较新的RStudio可以非常方便地实现代码标准化，写出的代码真是太标准了！嗯，首先就是要为代码加上合适的空格。举例来说，以前我会不假思索地写出`c(1,2,3,4,5,6)`这样的代码，连一个空格都懒得加。现在，我会写成`c(1, 2, 3, 4, 5, 6)`，每个逗号后面都加一个空格，这样代码就会更加易读。如果你还想了解更多代码标准，可以参见谷歌的R代码风格指南https://google.github.io/styleguide/Rguide.xml/。

我还收到了一些电子邮件，说我在网上获取的部分数据已经不存在了。国家冰球联盟已经决定使用一套全新的统计方法，所以我必须从头开始，重新做一遍那个例子。为了解决类似的问题，我把数据放到了GitHub上。

总而言之，为了给大家提供最好的工具，我尽了相当大的努力。另外，企业家马克·库班此前的一些评论在网络上引起了非常大的反响：

- “人工智能、深度学习、机器学习——如果你还不懂这些知识，那么一定要学习一下，不管你是做什么的。否则在3年之内，你就跟恐龙差不多了。”
- “我个人认为，在未来10年内，对文科专业人才的需求要超过对编程专业甚至工程专业人才的需求。因为当所有数据都呈现在面前时，我们就面临多种选择，这就需要以不同的视角来看待数据，以便得到各种不同的数据视图。所以需要更多思维更加开放的人才。”

这两条评论除了在博客圈内有一些交集之外，乍看上去彼此之间没有什么联系。但是仔细想一下，我认为他触到了我觉得自己应该写这本书的痛点。我坚信机器学习在某种程度上应该造福于大众。随着计算能力和信息可用性的不断提高，机器学习对于所有人来说都将是一种司空见惯的事情。但从另一方面看，机器学习还有一个问题，这个问题在现在和将来都会存在，那就是对结果的解释。如果你努力描述真阳性率和假阳性率时，对方一脸茫然，你应该怎么办？你怎样才能通过讲故事迅速启发听众？如果你做不到，请通知我，我非常愿意与你一起分享我的故事。

必须有人带头来做这些事情，并以此影响自己所在的组织。如果一个具有历史学或音乐鉴赏学位的人想做这些事，那就让他做吧。我每天都学习历史，它对我帮助巨大。库班的评论从多个方面使我更加确信，本书第1章最重要。如果你还没有向商业伙伴提出这个问题：“你想做些什么不一样的事情？”那么最好明天就去问。有太多人将太多努力花费在那些和组织及其决策完全无关的分析上。

本书内容

下面按章节给出本书对第1版做出的修改。

第1章重新制作了流程图，更正了一个无意中的输入错误，并新增了一些方法。

第2章改进了代码，并给出了更美观的图表，此外基本与第1版一致。

第3章改善并精简了代码。增加了多元自适应回归样条模型，这是我最喜欢的技术之一，它的效果非常好，可以处理非线性问题，而且易于解释。我将它作为基础模型，将其他模型作为“挑战者”，看看其他模型能否在性能上超过样条模型。

第4章不但介绍了回归模型中的特征选择技术，还包括了分类模型中的特征选择技术。

第5章梳理并精简了代码。

第6章增加了XGBOOST扩展包提供的流行技术，还增加了使用随机森林作为特征选择工具的技术。

第7章更新了一些深度学习方法的信息，并改进了使用H2O软件包的代码，包括超参数搜索技术。

第8章新增了使用随机森林进行无监督学习的方法。

第9章使用了新的数据集，新增了样本外预测的方法。

第10章新增了序列分析方法，我发现这种方法越来越重要，特别是在营销领域。

第11章属于全新内容，使用了若干个非常棒的软件包。

第12章添加了另外几年的气候数据，以及对几种不同因果关系测试方法的演示。

第13章增加了数据，改进了代码。

第14章也是新内容，帮助你在云上简单而又快速地获取R。

附录增加了新的数据处理方法。

准备工作

R是免费的开源软件，你只需从https://www.r-project.org/下载并安装即可。我强烈建议你从https://www.rstudio.com/products/RStudio/下载IDE和RStudio，当然，这一步不是必需的。

目标读者

本书的目标读者是数据科学家、数据分析师等专业人员。如果你具有使用R进行机器学习的工作经验，又想提高能力以成为机器学习领域的专家，那么本书也非常适合你。

排版约定

本书以不同文本样式区分不同种类的信息，下面列出并解释几种样式示例。

文本中的代码、数据库表名、文件夹名、文件名、文件扩展名、路径名、虚拟URL、用户输入和Twitter用户定位都表示为："可以在R的MASS包中找到该数据框，名为biopsy。"

所有命令行输入和输出都表示为：

```
> bestglm(Xy = biopsy.cv, IC="CV",
  CVArgs=list(Method="HTF", K=10,
  REP=1), family=binomial)
```

*新名词*和*重点词*会以楷体表示。显示器屏幕（比如菜单或对话框）上的词在文本中表示为："如果想下载新模块，我们可以使用**Files|Settings|Project Name|Project Interpreter**。"

警告或重要的注意事项。

提示或小技巧。

读者反馈

欢迎各位提出宝贵意见，请让我们知道你对本书的看法——喜欢什么或者不喜欢什么。读者反馈对我们非常重要，因为这可以帮助我们发现对大家最有帮助的主题。

要想提供反馈，只需登录"图灵社区"本书页面（http://www.ituring.com.cn/book/1989）并留言。

客户支持

如果您购买了我们出版的图书，我们将提供一系列服务来使您获得最大收益。

下载示例代码

你可以从"图灵社区"本书页面（http://www.ituring.com.cn/book/1989）下载书中示例代码。

文件下载结束之后，请确定使用以下软件的最新版本解压或提取文件：

- Windows系统：使用WinRAR或7-Zip
- Mac系统：使用Zipeg、iZip或UnRarX
- Linux系统：使用7-Zip或PeaZip

本书的代码包也保存在GitHub上：https://github.com/PacktPublishing/Mastering-Machine-learning-with-R-Second-Edition。https://github.com/PacktPublishing/，这个地址还提供了其他种类丰富的图

书和视频资料相关代码包，好好看一下吧！

勘误

尽管我们做了各种努力来保证内容的准确性，依然无法避免出现错误。如果你在书中发现文字或代码错误并告知我们，我们将非常感谢。通过勘误，有助提高其他读者的阅读体验，并帮助我们在本书的后续版本中做出改进。不管您发现什么错误，都可以通过“图灵社区”本书页面（http://www.ituring.com.cn/book/1989）告诉我们。一旦勘误通过确认，将显示在页面上的勘误表中。

反盗版

互联网上针对有版权资料的盗版行为一直存在，并逐步扩展到所有媒体。出版社非常重视对自己版权和许可的保护，如果您在互联网上发现对于我们工作的任何形式的非法复制行为，请立即将地址或网站名通知我们，我们会采取对策。

请联系ebook@turingbook.com并提供有盗版嫌疑的链接。

如果我们在作者保护和造福读者方面得到您的帮助，我们将非常感谢。

问题

对本书有任何疑问，都可以登录“图灵社区”本书页面（http://www.ituring.com.cn/book/1989），我们会尽最大努力解决问题。

电子书

如需购买本书电子版，请扫描以下二维码。

第1版前言

“诸事皆殚精竭虑者，终将一无所成。”

——腓特烈大帝

机器学习领域浩瀚无边，下面的引言对此进行了很好的概括：你面临的第一个问题就是令人眼花缭乱的学习算法，到底要用哪一个？现在已经有几千种算法，每年还会发布几百种新的算法（Pedro Domingo，2012）。如果要在正文中尝试涵盖所有算法，那就是不负责任了。因为按照腓特烈大帝的意思，我们将一事无成。

请牢记这个信条。本书目的就是为你在算法和业务方面打下坚实的基础，这会消除你的困惑，最重要的是，要使你充满信心地面对每一项机器学习任务，并且理解其他算法和主题。如果这本书对你的自我提升有明显帮助，那我认为这就是胜利。不要把本书当作一个目标，而要把它当作自我发现的途径。

R的世界同机器学习一样令人不知所措，R支持社区提供了多如牛毛的R包、博客、网站、讨论组以及水平各异的论文。这是很好的信息积累，并可能是R的最大优势。但我一直坚信，一个实体的最大优势也是其最大劣势。R庞大的知识社区可以轻易地使人无所适从或步入歧途。给我一个问题和10名R程序员，会得到解决这个问题的10种不同的代码编写方式。我在每一章都会尽力找到使用R进行数据理解、数据准备和数据建模的关键要素。我绝对算不上是R编程专家，但要再次强调，我会从打好地基开始。

燃起我写这本书的热情的另一个原因，是几年前发生的一件事。我的团队中有一个负责数据库管理的IT合同工。当时，我们一边走一边聊着大数据之类的话题，他说他买了两本书，一本关于使用R进行机器学习，另一本则使用的是Python。他称自己可以完成所有编程工作，但是完全不懂其中的统计学知识。写作本书的过程中，这场谈话一直萦绕在我的脑海里。技术、理论与实践的平衡一直是一项有挑战性的工作。肯定有人可以将每章中的理论单独写一本书，或许已经有人做到了。我用一种勉强称得上是启发式的方法来判断一个公式或一项技术是否有用，比如对我或读者在与团队成员或公司老板讨论时有所帮助。如果我认为有用，就会尽力提供必要的细节。

我特意对实际使用的数据集做了处理，使它们的规模既大到足以使用，又小到足以获取知识而不至于迷失。本书不是关于大数据的，但没关系，书中讲到的方法和概念完全可以扩展到大数

据方面。

简言之，本书对很多人群都具有意义，不论是试图理解和表述机器学习算法的IT精英，还是想在分析中发挥R的强大威力的统计学大师。尽管有些人同时精通IT技术和统计学，但他们在本书中仍然可以发现一些有用的窍门和技巧。

定义机器学习

机器学习已经无处不在！它可以用于网页搜索、垃圾邮件过滤、推荐引擎、医疗诊断、广告投放、欺诈检测、信用评分，甚至会用在自动驾驶汽车上。公路已经相当危险了，人工智能汽车每跑100英里（约160千米）就要用CTRL+ALT+DEL来重启，它们在高速公路和辅路上漫无目的地行驶，想想就令人害怕。好吧，我跑题了。

恰当地定义我们正在讨论的事物一直都很重要，机器学习也不例外。Machinelearningmastery.com这个网站用了一整页来讨论这个问题，并提供了很好的背景资料。它提供了一个简洁的、可接受的、可操作的定义，只有一句话："机器学习是使用数据对模型进行的训练，它针对某种性能指标形成决策。"

请记住这个定义。为进行机器学习，我们会有几个要求：第一，需要数据；第二，确实存在一个模式，也就是说，通过训练数据中的已知输入值，可以基于没有用于训练模型的数据做预测或决策，这就是机器学习中的泛化；第三，需要某种性能指标，以衡量学习/泛化的结果，比如均方误差、精确度或其他指标。本书会介绍几种性能指标。

在机器学习的世界中，我发现的趣事之一就是描述数据和流程的语言的变化。说到这儿，我忍不住要引用哲学家乔治・卡林的一段话：

> "没人告诉过我这件事，也没人问我是否同意，它就这么发生了。厕纸变成了卫生纸，胶鞋变成了跑步鞋，假牙变成了牙齿矫正器，吃药变成了药物治疗，问讯处变成了查号服务，垃圾场变成了填埋地，撞车变成了交通事故，局部多云变成了局部晴朗，汽车旅馆变成了汽车客栈，房车变成了活动房屋，二手车变成了曾被拥有的运输工具，客房服务变成了客房餐饮，便秘变成了偶发性不规律。"
>
> ——乔治・卡林，哲学家、喜剧演员

当我初入机器学习殿堂时，使用的是有因变量和自变量的数据集，建立模型的目标是找到最佳拟合。现在呢？我要对实例和输入特征做标记，然后进一步选择加工，最后生成用以学习模型的特征空间。当这一切都完成之后，以前我要做的是查看模型参数，而现在要检查权重。

我要告诉你，我会交替使用这些新旧名词，以后也会一直这样。机器学习的纯粹主义者可能会因此而诅咒我，但我不认为这样会造成什么严重的问题。

机器学习注意事项

在我们开启香槟，从此高枕无忧地认为机器学习会解决所有社会问题之前，还有一些相当重要的事情——注意事项。在实际工作中，要时刻将其记在心间，前事不忘，后事之师。

失败的特征工程

仅靠堆砌数据来解决问题是不够的，不管数据量有多大。这显而易见，我有过亲身经历，也见过其他人步入这个误区。商业领袖们天真地认为，只要提供巨量原始数据，再加上机器学习应有的魔力，就能解决一切问题。我之所以在第1章重点阐述如何限定业务问题和领导期望，以上就是原因之一。

除非数据来自于精心设计的实验，或者已经进行了预处理，否则原始观测数据几乎不可能直接用来建模。在任何一个项目中，实际花在建模上的时间都非常少。最需要花费时间的环节是特征工程：数据收集、数据集成、数据清洗和数据理解。在本书的练习中，我估计与建模相比，90%的时间要花在上述环节的编码工作上，这还是在大多数数据集都非常小且易于获取的情况下。在我的实际工作中，使用SAS时，99%的时间用在了PROC SQL上，只有1%的时间用在PROC GENMOD、PROC LOGISTIC或Enterprise Miner上。

对于特征工程，人们有两种观点。一种认为专业知识必不可少（我认同这一观点），另一种认为机器学习算法可以自动完成特征选择/构建的大部分工作。一些创业公司声称这是完全可行的。（我曾经和几个家伙聊过，谈理论时他们滔滔不绝，一旦涉及具体细节就闭口不言了。）假设你有几百个候选特征（自变量），进行自动特征选择的方法是计算单变量信息值。一个特征在孤立状态下会表现为完全不相关，但与另一个特征组合起来就可能变得非常重要。为解决这个问题，需要生成无数的特征组合。这必然会带来一个潜在的问题——计算时间和成本将大幅提高，并且可能造成模型的过拟合。说到过拟合，我们会在下一个注意事项中继续讨论。

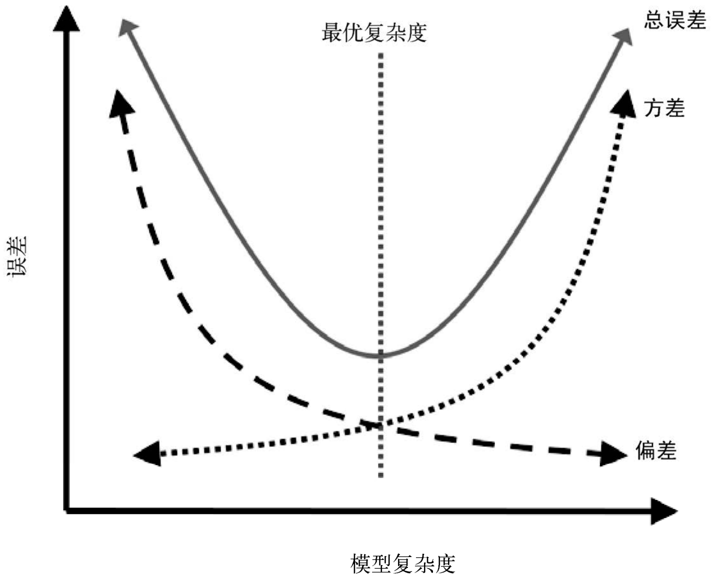
过拟合与欠拟合

当模型的泛化效果不佳时，就会表现出过拟合。如果你使用训练数据达到了95%的分类精确度，但使用另一组数据测试时，精确度却下降到了50%，那么模型的方差就太高了。如果训练数据的精确度为60%，测试数据的精确度为59%，那么模型的方差很小，但是偏差太大。这种偏差与方差之间的权衡是机器学习和模型复杂度的一个基本问题。

让我们先看看定义。偏差是模型的预测值或预测水平与训练数据中的实际值或实际水平之间的差别。方差是训练数据集的预测值或预测水平相对于其他数据集的预测值或预测水平的离散程度。当然，我们的目标是使总体误差（偏差+方差）最小，但这与模型复杂度又有什么关系呢？

为了说明其中的关系，假设要进行一项预测，并用训练数据建立一个简单的线性模型。这个

模型非常简单，所以具有高偏差；另一方面，训练数据和测试数据之间的方差却很小。如果我们在线性模型中加入多项式或者建立决策树，模型会变得更复杂，偏差会减小。但偏差减小的同时，模型的方差会扩大，泛化能力会降低。你可以在下图中看到这个现象。所有机器学习项目都应该尽力达到偏差和方差之间的最佳平衡点，这说起来容易，做起来很难。



我们会在其他章节讨论解决这个问题和优化模型复杂度的方法，包括交叉验证（第2~7章）和正则化（第4章）。

因果关系

相关性不等于因果关系，这一点应该已经广为人知，不需多费唇舌。但果真如此吗？现实世界中，明显有人依然搞不清相关性和因果关系的区别。所以，我们必须牢记并且坚定地告诉他人，算法基于观测数据而不是实验数据，不管从机器学习中得到多么完美的相关性，都不能胜过从正确的实验中得到的结论。正如佩德罗·多明戈斯教授所说：

> “如果发现人们在超市经常同时购买啤酒和纸尿裤，那么把啤酒放在纸尿裤旁边可能会提高销量。但如果没有经过实验验证，这个结论就站不住脚。”
>
> ——佩德罗·多明戈斯，2012

第11章会使用一种来自计量经济学的方法在时间序列中探索因果关系，讨论一个情感和政治敏感性的问题。

我就不再啰唆了，下面开始用R玩转机器学习吧！如果你对于R编程完全是个门外汉，我建议你跳过前面的内容，直接学习附录中的R使用方法。不管你从哪里开始阅读，请记住本书探讨的是掌握机器学习的过程，而不是要达到某个目标。只要我们在这个领域内辛勤耕耘，就会一直有令人惊喜的新事物值得我们探索。所以，我非常希望收到你的评论、想法、建议、抱怨和牢骚。就像印第安苏族勇士的口号一样："共同前进!"

本书内容

第1章说明机器学习不仅仅是写代码。为了使你的工作在业界具有持久的影响，我们介绍一个经过考验的流程，使你有个好的开始并走向成功。

第2章为学习支持向量机和梯度推进等高级方法打下坚实基础。没有比最小二乘线性回归更基础的方法了。

第3章讨论如何使用逻辑斯蒂回归与判别分析来预测分类结果。

第4章介绍正则化技术，帮助提高模型的预测能力和可理解性。特征选择是机器学习中最关键、最有挑战性的部分。

第5章开始研究更高级的非线性技术。机器学习的真正威力将揭开面纱。

第6章介绍几种机器学习领域内具有最强预测能力的技术，特别是对于分类问题而言。单决策树将与更高级的随机森林和提升树一起讨论。

第7章介绍一些当前应用中的最激动人心的机器学习技术。神经网络的灵感来自于大脑的工作原理，它将与其最近的高级分支——深度学习一同接受检验。

第8章开始涉及无监督学习，它的目标不是做出预测，而是把重点放在发现观测数据中的隐含结构上。我们将讨论3种聚类方法：层次聚类、K均值和围绕中心的划分（PAM）。

第9章继续研究无监督学习方法。主成分分析用来发现特征中的隐含结构，一旦发现其中的结构，新的特征将用于监督学习。

第10章介绍用来提高销量、检查欺诈和增进健康的技术。你将学习食品杂货店对于购买习惯的购物篮分析，然后研究如何在网站评估的基础上建立推荐引擎。

第11章讨论单变量预测模型、二元回归模型和格兰杰因果关系模型，还包括一个关于碳排放和气候变化的分析。

第12章展示一个定量文本挖掘框架以及如何建立主题模型。伴随着时间序列，数据世界包含着文本形式的海量数据。既然如此多的数据都是文本形式，那么懂得如何对文本数据进行处理、

编程和分析就显得特别重要。

附录介绍R的语法及其强大功能。R有一个陡峭的学习曲线，一旦你熟练掌握，就会发现对于数据准备和机器学习来说，R的威力有多么强大。

准备工作

R是免费的开源软件，你只需从https://www.r-project.org/下载并安装即可。我强烈建议你从https://www.rstudio.com/products/RStudio/下载IDE和RStudio，当然，这一步不是必需的。

目标读者

本书的目标读者是数据科学家、数据分析师等专业人员。如果你具有使用R进行机器学习的工作经验，又想提高能力以成为机器学习领域的专家，那么本书也非常适合你。

排版约定

本书以不同文本样式区分不同种类的信息，下面列出并解释几种样式示例。

文本中的代码、数据库表名、文件夹名、文件名、文件扩展名、路径名、虚拟URL、用户输入和Twitter用户定位都表示为："可以在R的MASS包中找到该数据框，名为biopsy。"

所有命令行输入和输出都表示为：

```
cor(x1, y1) #correlation of x1 and y1
[1] 0.8164205

> cor(x2, y1) #correlation of x2 and y2

[1] 0.8164205
```

*新名词*和*重点词*会以楷体表示。显示器屏幕（比如菜单或对话框）上的词在文本中表示为："如果想下载新模块，我们可以使用**Files|Settings|Project Name|Project Interpreter**。"

警告或重要的注意事项。

提示或小技巧。

读者反馈

欢迎各位提出宝贵意见，请让我们知道你对本书的看法——喜欢什么或者不喜欢什么。读者反馈对我们非常重要，因为这可以帮助我们发现对大家最有帮助的主题。

要想提供反馈，只需登录http://www.packtpub.com本书页面并留言。

客户支持

如果您购买了我们出版的图书，我们将提供一系列服务来使您获得最大收益。

下载示例代码

你可以从http://www.packtpub.com本书页面下载书中示例代码。

文件下载结束之后，请确定使用以下软件的最新版本解压或提取文件：

- Windows系统：使用WinRAR或7-Zip
- Mac系统：使用Zipeg、iZip或UnRarX
- Linux系统：使用7-Zip或PeaZip

https://github.com/PacktPublishing/，这个地址还提供了其他种类丰富的图书和视频资料相关代码包，好好看一下吧！

勘误

尽管我们做了各种努力来保证内容的准确性，依然无法避免出现错误。如果你在书中发现文字或代码错误并告知我们，我们将非常感谢。通过勘误，有助提高其他读者的阅读体验，并帮助我们在本书的后续版本中做出改进。不管您发现什么错误，都可以通过http://www.packtpub.com/submit-errata）告诉我们。一旦勘误通过确认，将显示在页面上的勘误表中。

反盗版

互联网上针对有版权资料的盗版行为一直存在，并逐步扩展到所有媒体。出版社非常重视对自己版权和许可的保护，如果您在互联网上发现对于我们工作的任何形式的非法复制行为，请立即将地址或网站名通知我们，我们会采取对策。

如果我们在作者保护和造福读者方面得到您的帮助，我们将非常感谢。

问题

对本书有任何疑问，都可以通过questions@packtpub.com联系我们，我们会尽最大努力解决问题。

目　录

第 1 章

成功之路

1

“如果你不知道要去哪里，就只能随波逐流，无所适从。”

——刘易斯 · 卡罗尔

“如果你不能将要做的事情描述成一个流程，那么你就不知道自己在做什么。”

——爱德华兹 · 戴明

乍一看，这一章跟机器学习没有什么关系，但实际上本章内容对于机器学习非常重要（特别是对于机器学习的实施以及由此造成的改变）。不管我们如何定义成功，最聪明的人、最好的软件和最好的算法都不能确保其实现。

在大多数（即便不是全部）项目中，成功解决问题或改进决策的关键因素不是算法，而是沟通能力和影响力之类的非定量的软技能。很多人认为其中的问题在于，我们很难量化这些软技能的效果。一般来说，人们遇到不想做的事情都会止步不前。别忘了，爆红的电视喜剧《生活大爆炸》就是这么拍的。所以，本章目的是使你走向成功，意在提供一个流程，至少是一个灵活的流程，使你成为一位**变革推动者**：一个不靠位高权重以势压人，而是具有真知灼见并能付诸实施的人。我们将集中讨论**跨行业数据挖掘标准流程**（**Cross-Industry Standard Process for Data Mining，CRISP-DM**），这可能是最著名也是最受重视的项目分析方法。即使你使用的是其他成熟方法或专有技术，也可以在本章有所收获。

我可以毫不犹豫地说，事情说起来容易，做起来难。对于本章内容中的错误和遗漏，我感觉非常内疚和遗憾。希望你能凭能力和运气避免我在过去12年中受到的各种身心上的伤害。

最后，我会介绍一个流程图（快速指南），你可以使用它判断应当使用何种方法解决手头的问题。

1.1 流程

CRISP-DM流程本来是专门为数据挖掘设计的，但它非常灵活和全面，完全可以应用于任何项目分析，无论是预测性分析、数据科学还是机器学习项目。不要被长长的任务列表吓倒，因为你可以在流程实施过程中根据自己的判断对流程进行调整以适应实际情况。图1-1给出了这个流

程的可视化表示，以及可以使流程非常灵活的反馈回路。

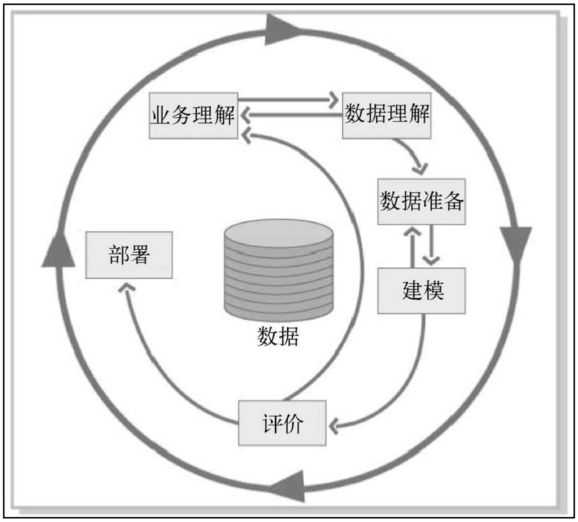


图1-1　CRISP-DM 1.0，循序渐进地进行数据挖掘

流程分以下6个阶段：

- 业务理解
- 数据理解
- 数据准备
- 建模
- 评价
- 部署

如果想查看包括所有任务和子任务的完整流程说明，请参考SPSS的文章“CRISP-DM 1.0, step-by-step data mining guide”（https://the-modeling-agency.com/crisp-dm.pdf）。

我会对流程中的每个步骤（包括其中的重要任务）进行说明，但这不是包含详细细节的说明书，而是更高层次的介绍。我不会跳过任何关键细节，但会重点介绍可以应用在任务上的技术。请记住，在后面的章节中我们将使用这些流程步骤，作为机器学习方法实际应用中的一个通用框架，特别是R语言实现。

1.2　业务理解

我们绝对不能低估流程的第一阶段对于最后成功的重要性，这是最基础的一个阶段，它的成败在很大程度上将决定项目其余部分的成败。本阶段的目的是确定业务需求，进而转化为分析目

标。这一阶段有以下4个任务：

(1) 确定业务目标
(2) 现状评估
(3) 确定分析目标
(4) 建立项目计划

1.2.1　确定业务目标

这一任务的关键是确定组织的目标并且限定问题的范围。一个好问题是：我们要做什么改变？这看上去是一个老掉牙的问题，但它确实可以使人们从分析的视角去思索到底想要什么，并且认识到需要做出的决策的根本目的。这个问题还可以防止你在调研中走得太远，或做一些不必要的工作。所以，最关键的一点就是确定**决策**。对于管理团队来说，决策的一个范例就是，对是否进行资源投入做出明确的选择。当然，选择不做任何改变也是一种决策。

这并不意味着如果选择没有彻底明确，项目就不能开始。有些时候，问题不存在或不能清楚定义，正如美国前国防部长唐纳德·拉姆斯菲尔德所说，这是“已知的未知”。实际上，问题经常没有得到清楚定义，项目的主要目标就是更加深入地理解问题并提出假设。又如拉姆斯菲尔德部长所说，还有“未知的未知”，也就是说你根本不知道自己不知道什么。但是，在没有定义清楚的问题中，我们可以根据基于各种假设所产出结果的资源投入，来理解接下来将发生什么。

在这一任务中，还需要重视对期望的管理。完美的数据是不存在的——不管其深度与广度如何。现在要基于你的专业知识说明什么是可行的，而不是做出保证。

我建议完成这一任务之后，要得到两个成果。第一个就是任务说明。这可不是单位中的琐碎冗长的任务说明，而是你自己的，更进一步说，是经过项目负责人确认过的任务说明。我是从多年的军旅生涯中悟到这一点的，我可以长篇大论地说明为什么它非常有效，但那是以后的事。我们可以认为，当没有明确的方向和指导时，任务说明（或者随便你想叫它什么）是所有利益相关者的统一声明，可以防止需求范围蔓延。它包括以下几部分。

- **谁**：你、你的团队或者项目名称，所有人都喜欢一个酷酷的项目名，例如“蝰蛇行动”“融合”，等等。
- **什么**：你要进行的任务，例如，实施机器学习。
- **何时**：最后期限。
- **何地**：可以是地理意义上的，也可以是功能、部门、自发项目等。
- **为什么**：项目的目的，也就是业务目标。

第二个成果就是要尽可能明确成功的定义。达到哪些条件才算是成功？要帮助团队或负责人清晰描述出你所理解的成功，然后你的工作就是将其转化为建模需求。

1.2.2　现状评估

这项任务用于在项目计划中收集信息，包括可用资源、限制条件和假设；还要识别风险，并做出应急方案。多说一句，在这个阶段，还要确定关键的利益相关者，也就是要受到未来决策影响的人。

这里有几点需要注意。当检查可用资源时，不要忘了将过去和现在的项目记录都看一遍。要特别关注组织中曾经或正在处理相同问题的人，要将你的工作和他们的工作结合起来。不要忘了列举风险，时间、人员和费用都会产生风险。尽你的最大努力建立一个利益相关者列表，包括那些影响项目的人和被项目影响的人，确定这些人是谁，如何影响决策或被决策影响。这项工作完成之后，就要和项目负责人一起建立一个与这些利益相关者的沟通计划。

1.2.3　确定分析目标

在这一步，你需要将业务目标转化为技术需求。这需要将“确定业务目标”任务中的成功标准转化为技术上的成功标准，此时可能要引入均方根误差或预测准确度水平等标准。

1.2.4　建立项目计划

这一步的任务是基于目前收集到的所有信息建立一个有效的项目计划。不管你使用什么技术，甘特图或其他图表都可以，一定要使其成为沟通计划的一部分。要使大多数利益相关者了解这个计划，并根据实际情况定期更新。

1.3　数据理解

在经过了虽然痛苦但却至关重要的第一阶段之后，你可以着手于数据工作了。这个阶段的任务如下：

(1) 数据收集
(2) 数据描述
(3) 数据探索
(4) 数据质量校验

这一步是**ETL**（**数据抽取、转化和加载，Extrack, Transform, Load**）的典型案例，也有几个需要注意的地方。你需要做初步判定，确认目前可用的数据能够满足你的分析要求。当你使用可视化或其他方法进行数据探索时，需要确定变量是否稀疏以及数据缺失程度，从而确定要使用的机器学习方法，并确认对缺失数据的填补是否必要和可行。

数据质量的校验是非常重要的。需要花费一些时间来搞清楚数据是由谁收集的，如何收集的，

甚至还要搞清为什么要收集数据。你经常会遇到数据收集不完整的情况，意外的IT问题会导致数据错误，业务规则也会按计划进行调整。这在时间序列分析中非常重要，决定数据分类的业务规则会经常随时间变化。最后，从这个阶段开始对程序代码进行归档是个好主意。作为归档过程的一部分，如果数据字典不可用，那么为了防止潜在的严重问题，请一定要做一个数据字典。

1.4 数据准备

差不多了！这个阶段包括下面5个任务：

(1) 数据选择
(2) 数据清洗
(3) 数据构建
(4) 数据整合
(5) 数据格式化

这些任务不需太多解释，其目的就是准备好数据以输入算法，包括数据合并、特征工程、数据转换等。如果需要填补缺失数据，也要在这个阶段完成。尤其是使用R，则要注意输出结果需要如何标记。如果输出变量（响应变量）是Yes/No的形式，那么在有些程序包中是不被支持的，需要进行数据转换，否则就没有1/0变量。在这个阶段中，如果需要，还要将数据分成不同的集合：训练集、测试集或验证集。这个阶段的工作极其繁重，但是很多过来人会告诉你，这就是你脱颖而出的机会。做好这些工作，我们就可以得到丰厚的回报。

1.5 建模

在这个阶段，你之前所做的一切工作都该有个结果了——或者挥拳庆祝，或者抱头痛哭。别在意，如果这个工作那么简单，那不是谁都能做了吗？本阶段的任务包括：

(1) 选择建模技术
(2) 设计检验方法
(3) 建立模型
(4) 评估模型

奇怪的是，这个流程阶段中需要注意的事情都是你已经考虑过的和准备好的。在第一阶段，你对如何进行建模总会有一点概念。请记住，这是一个灵活的、可迭代的流程，而不是像机组人员备忘录那样严格的线性流图。

本章后面的快速指南可以帮助你正确选择建模技术。检验设计指的是如何建立你的测试数据集和训练数据集，以及如何使用交叉验证。在数据准备阶段就需要考虑这些事情了。

模型评估需要将模型与在业务理解阶段建立的成功标准进行对比，比如均方根误差、提升度、ROC曲线等。

1.6　评价

在评价阶段，主要目的是确认已经完成的工作和选择的模型是否符合业务目标。问一下自己和他人，我们达到项目成功的要求了吗？看看Netflix这一反面教材。我相信你一定知道，Netflix悬赏100万美元寻找团队，开发最好的、具有最低均方根误差的推荐算法。但是，Netflix并没有实施这个算法，因为算法获得的精确度提升根本配不上实施成本。请永远记住奥卡姆剃刀原理。评价阶段至少包括以下任务：

(1) 评价结果
(2) 回顾过程
(3) 确定下一步

回顾过程时，非常有必要使工作结果获得管理层的认可，并与其他利益相关者进行沟通，以取得他们的支持。当然，在流程的前几个阶段也是如此。如果你想成为变革推动者，下一个步骤就是确保你已经回答利益相关者头脑中的这几个问题：**是什么**？**要什么**？**现在要做什么**？如果你能用前面阶段中产生的决策来说服他们现在要做什么，那你就成功了。

1.7　部署

如果直到现在所有事情都按照计划进行，那么只需按动一下开关，你的模型就会运转。假设情况并非如此简单，那么这个阶段要进行以下任务：

(1) 按计划部署
(2) 监测与维护
(3) 完成总结报告
(4) 项目回顾

在部署、监测和维护工作展开之后，对于你和你的团队来说，最重要的事情就是完成一份完整详尽的总结报告。报告应该包括一份白皮书和一份幻灯片简介。我要解释一下，我没有按照自己的意愿将工作成果放在白皮书中，因为我和军方签有合同，军方强烈要求使用PowerPoint幻灯片。但是，幻灯片很可能是某些人出于自己的目的来应付你的，华而不实，空话连篇。相信我，白皮书不会有这个问题，因为它是你的工作成果和信念的延续。如果你所在的组织坚持使用PowerPoint，那么你可以使用它向利益相关者做简单的介绍，但要使用白皮书作为文档记录和预读资料。在R中，生成白皮书的标准过程是使用knitr和LaTex包。

现在，我们已经讨论了所有重要阶段，你可以大展身手了。但是不管你以正式方式还是非正式方式推进流程，其中总要包括以下几点：

- 计划做什么？
- 实际做了什么？
- 为什么做这些工作，或为什么不做？
- 以后的项目中还需要做什么支持工作？
- 以后的项目中要进行哪些改进？
- 确定行动计划，保证支持和改进工作顺利进行。

综上所述，我们完成了对CRISP-DM流程的介绍。这个流程提供了一个综合而又灵活的框架，以保证项目成功实施，并使你成为那个推进变革的人。

1.8 算法流程图

本节的目的是建立一个工具，它不但可以帮助你选择合适的建模技术，而且可以帮助你更加深入地思考问题，你也可以用它和项目团队或项目负责人一起做出问题的框架。这种使用流程图的技术并不复杂，足以让你开始工作了。它还包括了一些本书没有涉及的技术。

图1-2启动了一个流程，通过这个流程你可以选择可能使用的建模技术。你只需回答问题，它会指引你进入图1-3~图1-6。

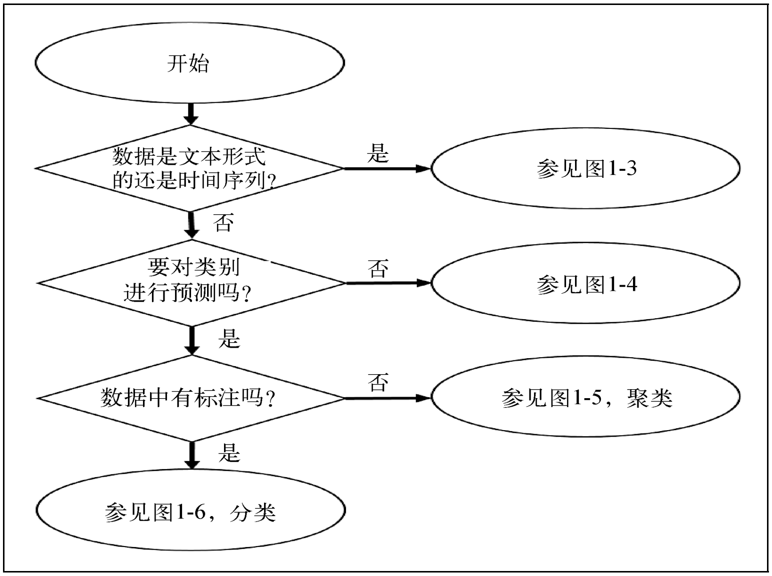


图1-2

如果数据是文本形式的或时间序列形式的，那么你应该采用图1-3中的流程。

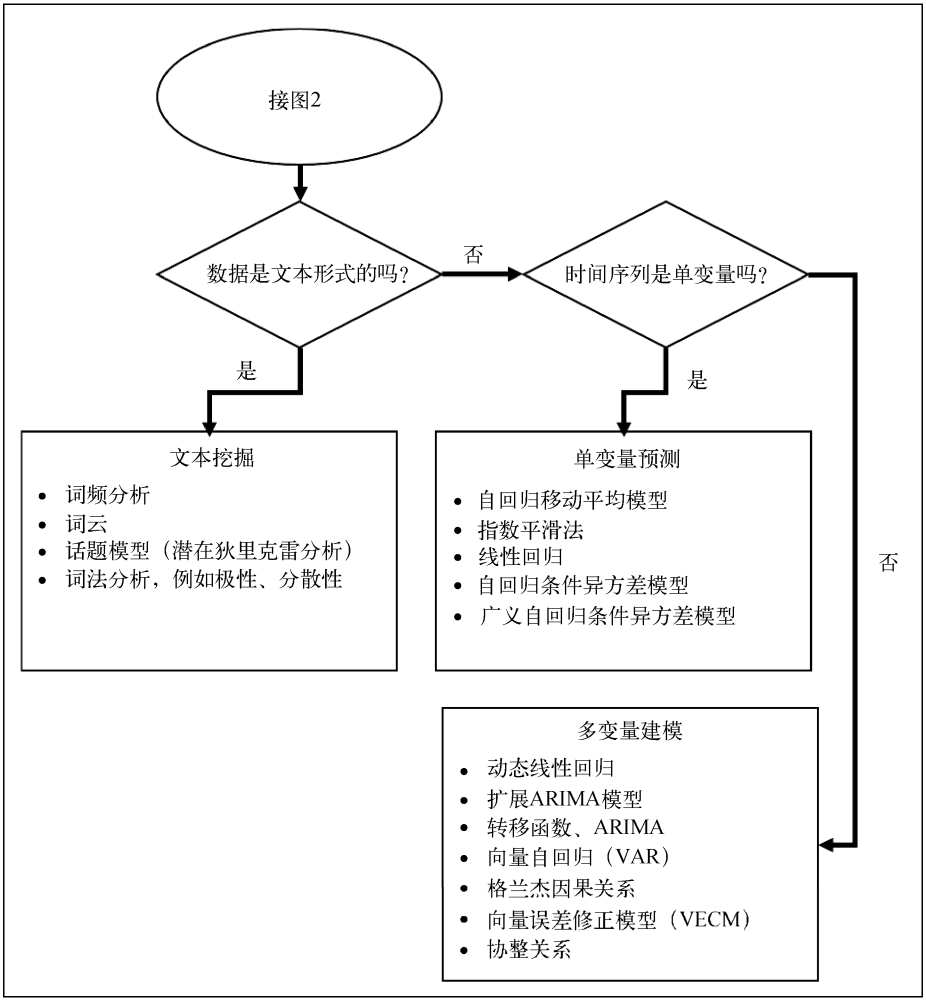


图1-3

图1-4所示的这个算法分支中，你不需要文本数据和时间序列数据，也不需要预测分类。所以，你的目的应该是做出推荐、理解关联规则或预测出一个数值量。

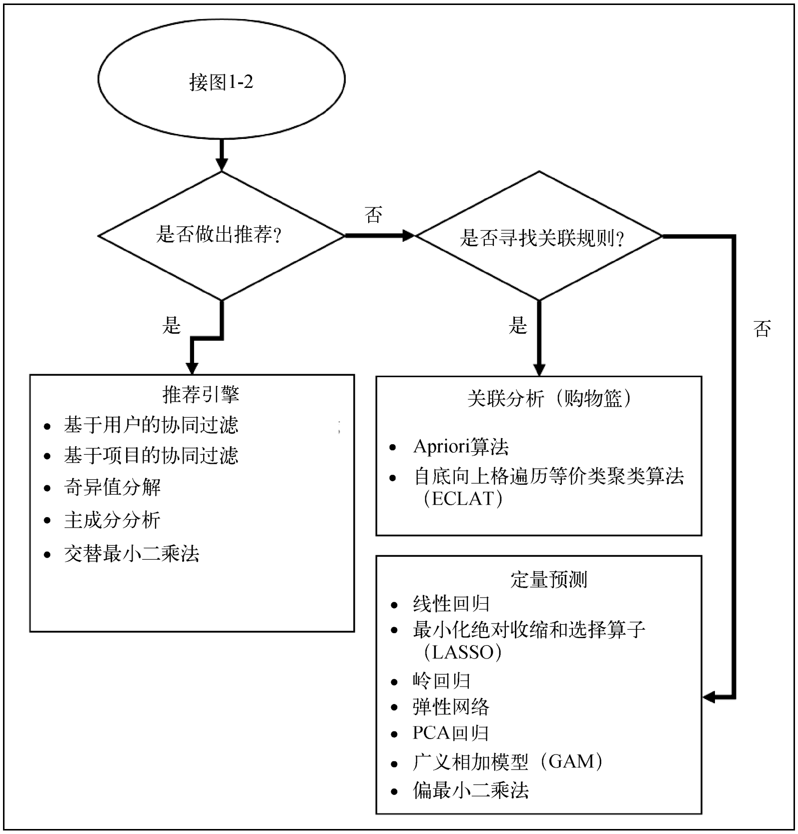


图1-4

要进入图1-5，你的数据不能是文本形式的或时间序列。你的目的是对数据进行分类，但分类结果不用标记，这就要使用聚类方法。

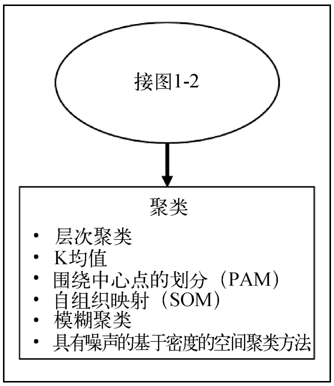


图1-5

如果想对数据分类并进行标记，则使用分类技术，如图1-6所示。

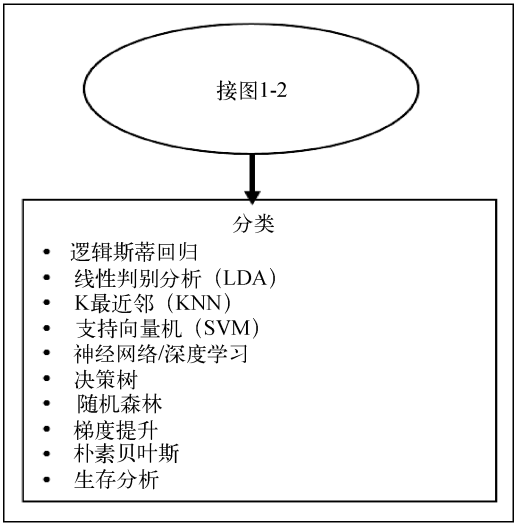


图1-6

1.9 小结

本章介绍如何使你自己和你的团队在承担的项目中取得成功。我们介绍了CRISP-DM流程，这是一种灵活而又全面的框架，其目的是提高沟通能力和影响力等软技能。我们列举了流程的每个阶段及其任务，还详细介绍了一些技术和注意事项，以保证流程顺利执行。如果你能认真遵循流程，那么在任何组织中，你都可以成为积极的变革推动者。

本章的另一部分内容是关于算法流程图的，这是一份快速指南，可以帮助你确定合适的技术，以解决业务问题。基础已经打好了，我们的下一步工作就是使用机器学习技术解决实际问题。

第 2 章

线性回归：机器学习基础技术

“有人总想在这项竞技中找到根本不存在的东西，但橄榄球世界里只有两件事——拦截与抢断。”

——文斯 · 隆巴迪，橄榄球名人堂教练

我们非常有必要从一项简单但又特别有效的技术开始，这项技术已经应用了很长时间，它就是**线性回归**。我记得阿尔伯特 · 爱因斯坦曾经说过，事情应该尽可能简单，直到不能再简单为止。这真是至理名言，也是我们开发机器学习算法时应该遵循的经验法则。线性回归使用**最小二乘法**预测定量的结果，想想我们随后将讨论的其他技术，真的没有比久经考验的线性回归更简单的模型了。实际上，线性回归是我们后面要讨论的所有方法的基础，很多方法仅是线性回归的扩展。坦白地说，如果你能掌握线性回归方法，那么本书的其余部分就易如反掌。因此，在我们成为机器学习专家的道路上，线性回归是一个非常好的起点。

本章包含一些入门级介绍，如果你是这方面的专家，可以跳过这些内容，直接进入下一主题。否则，请确保你完全理解了线性回归，然后才能开始学习那些更复杂的机器学习方法。你会发现很多项目仅靠随后讨论的技术就可以完成。线性回归可能是最容易向客户解释的模型了，大多数客户都能大致理解**R方**（**R-squared**）的意义，很多人可以理解得更深入，对变量贡献、**共线性**等概念都能接受。

2.1　单变量回归

我们先从一个简单的对定量型响应变量的预测开始。令这个响应变量为Y，还有一个预测变量x，假设Y与x具有线性关系，那么这个预测模型可以表示为$Y = B_0 + B_1x + e$。我们规定，Y的预测值是一个函数，等于B_0（截距）加上B_1（斜率）乘以x再加上一个误差项e。最小二乘法选择模型参数，使预测值y和实际值Y的**残差平方和**（**RSS**）最小。举个简单的例子，假设我们有两个实际值Y_1和Y_2，分别等于10和20，两个预测值y_1和y_2分别等于12和18。要计算RSS，只需把它们的差的平方相加：$\text{RSS} = (Y_1 - y_1)^2 + (Y_2 - y_2)^2$，再做一个简单的代入，可以得到：$(10 - 12)^2 + (20 - 18)^2 = 8$。

我曾经和一个一起进行精益六西格玛黑带培训的伙伴说过，线性规划中最重要的就是平方和；理解了平方和，其余就水到渠成了。从某种程度来说，的确如此。

开始实际应用之前，我想提醒一下，当你看到有关突破性研究的报道时，先不要轻信，要有质疑的态度，因为媒体发表的结论可能未经验证。我们知道，对于R或其他相关软件，只要有输入，就会给出一个结果。但是，仅凭数学上有意义和很高的相关性以及漂亮的R方统计量，是不能认为结论正确的。

为了说明这个问题，请看R中著名的Anscombe数据集。它由统计学家弗朗西斯・安斯库姆（Francis Anscombe）建立，用来强调数据可视化和异常值在数据分析中的重要性。这个数据集有4对*X*变量和*Y*变量，它们具有相同的统计特性。但如果将其放在统计图中，就会看到一些极大的差异。我用这个数据集进行内部培训，还教育过那些只盯着统计量而不进行数据探索和假设检验的商业伙伴。如果你有同样的需求，那这个例子就是一个非常好的开始。这只是我们正式建模之前的一个小插曲。

```
    > #call up and explore the data

> data(anscombe)

> attach(anscombe)

> anscombe
   x1  x2  x3  x4    y1    y2    y3    y4
1  10  10  10   8  8.04  9.14  7.46  6.58
2   8   8   8   8  6.95  8.14  6.77  5.76
3  13  13  13   8  7.58  8.74 12.74  7.71
4   9   9   9   8  8.81  8.77  7.11  8.84
5  11  11  11   8  8.33  9.26  7.81  8.47
6  14  14  14   8  9.96  8.10  8.84  7.04
7   6   6   6   8  7.24  6.13  6.08  5.25
8   4   4   4  19  4.26  3.10  5.39 12.50
9  12  12  12   8 10.84  9.13  8.15  5.56
10  7   7   7   8  4.82  7.26  6.42  7.91
11  5   5   5   8  5.68  4.74  5.73  6.89
```

可以看到，每对变量都具有相同的相关系数0.816。前两对变量的相关系数如下：

```
    > cor(x1, y1) #correlation of x1 and y1
[1] 0.8164205

> cor(x2, y1) #correlation of x2 and y2

[1] 0.8164205
```

当我们画出这4对变量的统计图时，就能看出问题了，这就是Anscombe这个数据集的设计目的。如下所示：

```
> par(mfrow = c(2,2)) #create a 2x2 grid for
  plotting

> plot(x1, y1, main = "Plot 1")

> plot(x2, y2, main = "Plot 2")

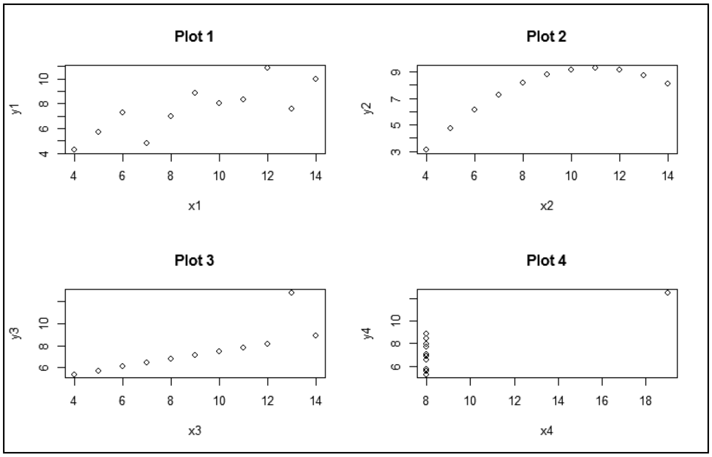
> plot(x3, y3, main = "Plot 3")

> plot(x4, y4, main = "Plot 4")
```

下载示例代码

可以通过“图灵社区”本书页面（http://www.ituring.com.cn/book/1989）下载书中示例代码。

上述代码输出如下。



可以看到，Plot 1中呈现的是真正的线性关系，Plot 2是一条曲线，Plot 3有一个危险的离群点，Plot 4则完全被离群点“拐跑了”。看到了吧，这就是一则警世恒言，说明了仅看相关性有多么危险。

业务理解

我们的第一个例子重点关注的是预测怀俄明州蛇河流域的水量（以英寸为计量单位），它是年度降雪含水量的一个函数。这项预测有助于管理水流量和蓄水量，因为蛇河是美国西部几个州农牧场灌溉用水的主要来源。数据集snake可以在alr3包中找到（注意，alr表示实用线性回归）：

```
> install.packages("alr3")
> library(alr3)
> data(snake)
> dim(snake)
[1] 17 2
```

```
> head(snake)
     X    Y
1 23.1 10.5
2 32.8 16.7
3 31.8 18.2
4 32.0 17.0
5 30.4 16.3
6 24.0 10.5
```

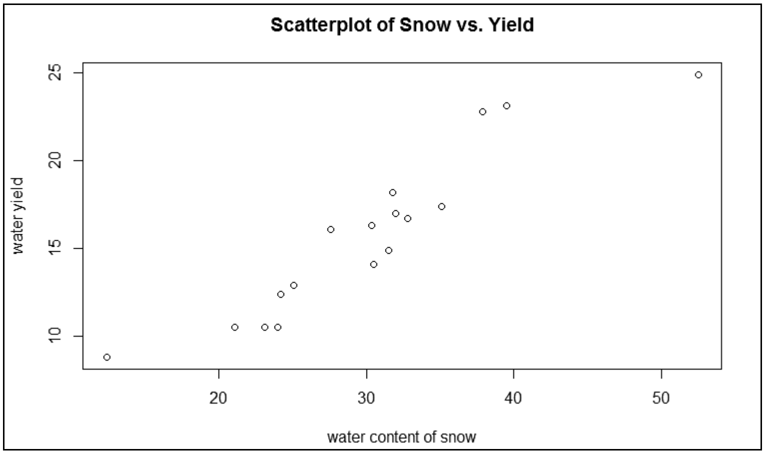
既然我们有了17行观测值，下面可以进行数据探索了。别急，先将*X*和*Y*换成有意义的变量名，如下所示：

```
    > names(snake) <- c("content", "yield")
> attach(snake) # attach data with new names
> head(snake)

    content yield
1      23.1  10.5
2      32.8  16.7
3      31.8  18.2
4      32.0  17.0
5      30.4  16.3
6      24.0  10.5

> plot(content, yield, xlab = "water content of
        snow", ylab = "water yield")
```

上述代码输出如下。



这张图很有意思，它的数据是线性的，因为被最前端和最后端的两个疑似离群点影响，有一点轻微的曲线形状。所以，有必要进行数据转换并删除无关观测值。

R使用`lm()`函数进行线性回归，`lm()`可以建立一个标准形式的回归模型fit = lm(*Y*~*X*)。建立模型之后，你可以对拟合模型使用各种函数，以检验自己的假设。代码如下：

```
> yield.fit <- lm(yield ~ content)

> summary(yield.fit)

Call:
lm(formula = yield ~ content)

Residuals:
        Min     1Q Median      3Q     Max
-2.1793 -1.5149 -0.3624 1.6276 3.1973

Coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.72538 1.54882  0.468 0.646
content     0.49808 0.04952 10.058 4.63e-08
     ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1

Residual standard error: 1.743 on 15 degrees of
      freedom
Multiple R-squared: 0.8709,     Adjusted R-squared:
       0.8623
F-statistic: 101.2 on 1 and 15 DF, p-value:
       4.632e-08
```

通过`summary()`函数，我们可以查看模型包含的一些项目，比如模型具体参数、关于残差的描述性统计量、系数、模型显著性代码、模型误差和拟合程度的摘要。现在，让我们重点关注对于相关系数这个参数的估计，看一下我们的预测变量是否具有显著的*p*值，以及整个模型的F检验是否具有显著的*p*值。请看参数估计，模型告诉我们，`yield`等于0.72538加上0.49808乘以`content`。可以确定，`content`每变动1个单位，`yield`会增加0.49808个单位。F统计量是用来检验原假设的，原假设认为模型的所有系数都是0。

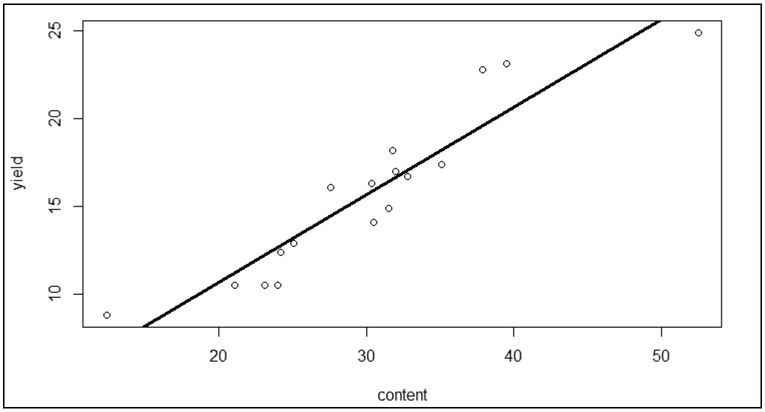
因为*p*值是高度显著的，所以我们可以拒绝原假设。接下来看`content`变量的t检验值，原假设认为它应该是0，我们又一次拒绝了原假设。此外，还可以看一下`Multiple R-squared`和`Adjusted R-squared`的值。我们将在多变量回归部分讨论`Adjusted R-squared`，所以先关注`Multiple R-squared`，它的值为0.8709。理论上，`Multiple R-squared`的取值范围在0和1之间，用来表示*X*和*Y*的相关程度。在本例中，它的意义是**水量**87%的方差可以被**降雪含水量**解释。顺便说一句，R平方项就是[*X*, *Y*]的相关系数的平方。

再回到散点图。可以用下面的代码为散点图加上一条由模型产生的最佳拟合直线。

```
> plot(content, yield)

> abline(yield.fit, lwd=3, col="red")
```

代码输出如下。



线性回归必须通过假设检验，其中的假设可以总结如下。

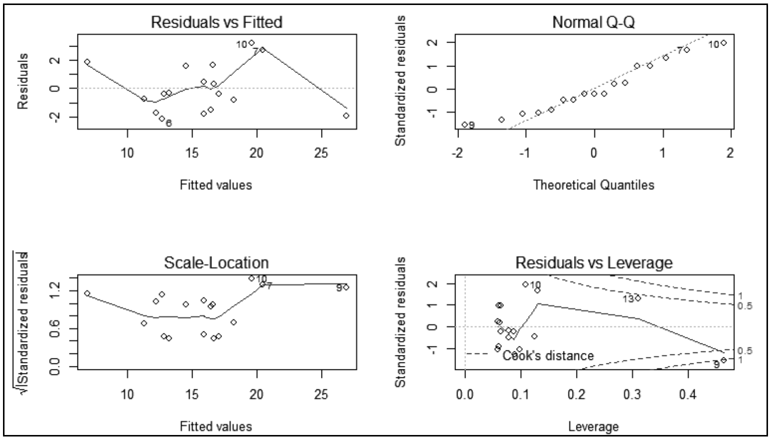
- **线性**：预测变量与响应变量之间的关系是线性的。如果线性关系不能清晰呈现，可以对变量*X*或*Y*进行数据转换（对数转换、多项式转换、指数转换等）以解决问题。
- **误差不相关**：在时间序列和面板数据中，$e_n = \text{beta}_{n-1}$是一个常见问题；如果误差是相关的，那么你就有可能建立一个非常不规范的模型。
- **同方差性**：误差是正态分布的，并具有相同的方差。这意味着对于不同的输入值，误差的方差是个固定值。如果违背了这个假设，参数估计就有可能产生偏差，导致对显著性的统计检验结果过高或者过低，从而得到错误的结论。这种情况就称为**异方差性**。
- **非共线性**：两个预测变量之间不存在线性关系，也就是说，特征之间不应该存在相关性。同样地，共线性也会导致估计偏差。
- **存在异常值**：异常值会严重影响参数估计。理想情况下，必须在使用线性回归拟合模型之前就除去异常值。正如我们在Anscombe数据集那个例子中看到的，异常值也会导致具有偏差的估计结果。

因为我们建立的单变量模型与时间不相关，所以只须关注线性和异方差性。在下一节中，其他假设会变得重要。对假设进行初步检验的最好方式就是画统计图。`plot()`函数结合线性模型，可以自动生成4张统计图，帮助我们进行假设检验。R会一次性生成这些图，你可以通过回车键进行切换。最好的方式是同时查看这4张图，我们通过以下方式实现：

```
> par(mfrow = c(2,2))
```

```
> plot(yield.fit)
```

上述代码输出如下。



左边的两张图供我们检查误差的同方差性和非线性。我们期望能发现某种模式，或者更重要的是，不存在任何模式。在样本大小只有17个观测值的情况下，不会看到什么明显的模式。误差的异方差性通常会表现为U形曲线或反U形曲线，也可能会紧密聚集在图的左侧，随着拟合值的增加逐渐变宽（漏斗形）。我们完全可以断定，模型明显没有违背同方差性假设。

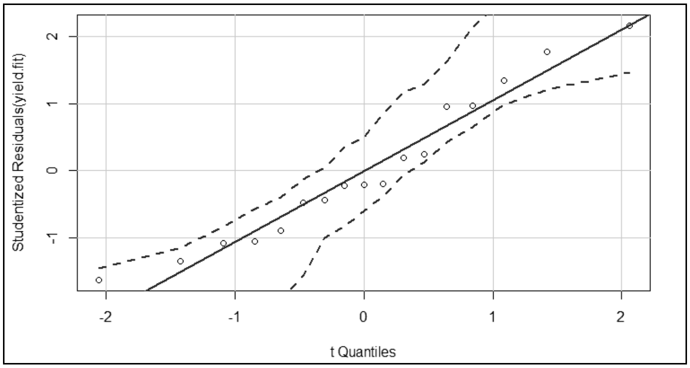
右上角的**正态Q-Q图**可以帮助我们确定残差是否服从正态分布。**分位数–分位数图**表示一个变量的分位数对应于另一个变量的分位数画出的图，从图中可以看出有离群点（第7、9、10个观测），这可能违反假设。**残差杠杆图**可以告诉我们哪个观测值（如果有）会对模型造成过度影响，换句话说，是否存在我们应该关注的异常值。鉴别强影响点的统计量是**库克距离**，一般认为，如果这个统计量的值大于1，就需要进行更深入的检查。

那么到底该如何进行呢？这既是科学，又是艺术。最简单的方法就是直接删除观测值，在本例中我们可以删除第9个观测值，然后重新构造模型。但更合理的选择是，转换预测变量或响应变量的形式。如果仅仅删除第9个观测值，那么第10个与第13个观测值就可能因为库克距离大于1而落到正常区间之外。我相信，这就是领域专家的用武之地。我无数次地发现，对异常值的研究和理解可以得到非常有价值的知识。当我们第一次查看前面的散点图时，我曾经指出几个疑似离群点，恰巧就是第9个和第13个观测值。作为一名分析师，非常重要的一点就是要经常与相关主题专家进行讨论，以弄清为什么会出现这种情况。是测量出现了错误？还是对这些观测值有更合理的解释？我当然给不出答案，但这肯定可以给你的组织带来更大价值。

这个话题先说到此，下面对现有模型做更深一步的研究，看看**正态Q-Q图**的更多细节。R在缺省的Q-Q图上没有提供置信区间，但仔细查看基础Q-Q图后，我们认为有必要检查置信区间。car包中的`qqPlot()`函数可以自动提供置信区间，因为car包与alr3包是一起加载的，所以可以用一行代码生成所需统计图，如下所示：

```
> qqPlot(yield.fit)
```

上述代码输出如下。



如图所示，残差服从正态分布。我认为，这可以使我们更有信心选择使用所有观测值拟合的模型。改进模型时，需要清晰理智的思考和判断。如果我们明确拒绝了误差正态分布假设，那么就应该考虑进行变量转换和删除观测值。

2.2　多变量线性回归

你可能正在扪心自问，在现实世界中，一个预测变量真的够用吗？这确实是一个好问题，而且一个变量在大多数情况下确实不够（时间序列通常是一个例外）。更可能的情况是，模型中不得不引入多个（如果不是许多）预测变量——或特征，机器学习中更喜欢使用这个名词。我们下面将讨论多变量线性回归，并开始一个新的商业案例。

2.2.1　业务理解

我们的主题还是水域保护和预测，使用alr3包中的另一个数据集，名字是water。2014年，美国南加利福尼亚严重的旱灾引起了很大关注，甚至连加州州长杰瑞·布朗都采取了行动，呼吁市民将用水量减少20%。在这个练习中，假设我们被加州政府委托预测水的可用性。提供给我们的数据包含43年的降雪量，收集自欧文斯山谷的6个不同地点。数据集中还有一个表示水的可用性的响应变量——加州毕绍普市附近的河川径流量，这些流量被引入欧文斯山谷的引水渠，最后流入洛杉矶引水渠。对径流量的精确预测可以使工程师、规划者和政策制定者更有效地制定水域保护措施。我们要建立的模型形式是$Y = B_0 + B_1x_1 + \cdots + B_nx_n + e$，预测变量（特征）可以有$1 \sim n$个。

2.2.2　数据理解和数据准备

为开始这一步，我们加载名为water的数据集，并用`str()`函数查看其结构。如下所示：

```
> data(water)

> str(water)
```

```
'data.frame':   43  obs. of  8 variables:
$ Year    : int  1948  1949 1950 1951 1952 1953 1954
      1955 1956 1957 ...
$ APMAM   : num  9.13  5.28 4.2 4.6 7.15 9.7 5.02 6.7
       10.5 9.1 ...
$ APSAB   : num 3.58 4.82 3.77 4.46 4.99 5.65 1.45
       7.44 5.85 6.13 ...
$ APSLAKE: num 3.91 5.2 3.67 3.93 4.88 4.91 1.77
       6.51 3.38 4.08 ...
$ OPBPC   : num 4.1 7.55 9.52 11.14 16.34 ...
$ OPRC    : num 7.43 11.11 12.2 15.15 20.05 ...
$ OPSLAKE: num 6.47 10.26 11.35 11.13 22.81 ...
$ BSAAM   : int 54235 67567 66161 68094 107080
       67594 65356 67909 92715 70024 ...
```

2

我们有8个特征，其中BSAAM是响应变量。观测值始于1943年，并不间断地进行了43年。因为我们不关心观测发生在哪一年，所以应该建立一个新的数据框，去掉年份向量。这做起来非常简单，用一行代码即可建立新的数据框，然后用`head()`函数检验结果是否正确。如下所示：

```
    > socal.water <- water[ ,-1] #new dataframe with
      the deletion of
      column 1

> head(socal.water)
  APMAM APSAB APSLAKE OPBPC  OPRC OPSLAKE  BSAAM
1  9.13  3.58    3.91  4.10  7.43    6.47  54235
2  5.28  4.82    5.20  7.55 11.11   10.26  67567
3  4.20  3.77    3.67  9.52 12.20   11.35  66161
4  4.60  4.46    3.93 11.14 15.15   11.13  68094
5  7.15  4.99    4.88 16.34 20.05   22.81 107080
6  9.70  5.65    4.91  8.88  8.15    7.41  67594
```

既然所有特征都是数值型的，那么就应该检查相关性方面的统计量，并绘出散点图矩阵。相关系数又称**Pearson's r**，可以用来测量两个变量之间线性相关性的强度和方向。这个统计量是一个数值，取值在–1和1之间，–1表示完全负相关，+1表示完全正相关。要计算相关系数，需要使用两个变量的协方差除以它们的标准差的乘积。前面说过，如果取相关系数的平方，就可以得到R方。

有很多方式可以做出相关性矩阵图。有些人喜欢做**热点图**，但是我强烈推荐使用corrplot包做图，它可以用很多方式表示相关系数之间的差异，包括椭圆、圆、正方形、数字、阴影、颜色和饼图。我更喜欢椭圆，当然也不反对你试试其他方式。下面加载corrplot包，使用基础的`cor()`函数建立一个相关性对象，然后看看结果。如下所示：

```
    > library(corrplot)

> water.cor <- cor(socal.water)

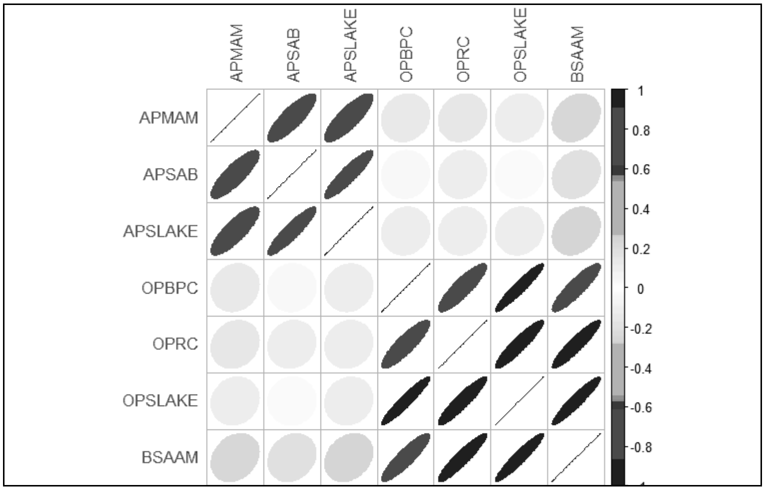
> water.cor
  APMAM       APSAB   APSLAKE       OPBPC
```

```
APMAM   1.0000000 0.82768637  0.81607595 0.12238567
APSAB   0.8276864 1.00000000  0.90030474 0.03954211
APSLAKE 0.8160760 0.90030474  1.00000000 0.09344773
OPBPC   0.1223857 0.03954211  0.09344773 1.00000000
OPRC    0.1544155 0.10563959  0.10638359 0.86470733
OPSLAKE 0.1075421 0.02961175  0.10058669 0.94334741
BSAAM   0.2385695 0.18329499  0.24934094 0.88574778
             OPRC    OPSLAKE       BSAAM
APMAM   0.1544155 0.10754212  0.2385695
APSAB   0.1056396 0.02961175  0.1832950
APSLAKE 0.1063836 0.10058669  0.2493409
OPBPC   0.8647073 0.94334741  0.8857478
OPRC    1.0000000 0.91914467  0.9196270
OPSLAKE 0.9191447 1.00000000  0.9384360
BSAAM   0.9196270 0.93843604  1.0000000
```

能看出什么？首先，响应变量与那些OP开头的特征高度正相关，与OPBPC的相关系数是0.8857，与OPRC的相关系数是0.9196，与OPSLAKE的相关系数是0.9384。还可以看出，AP开头的特征彼此之间高度相关，OP开头的也是如此。这意味着我们会遇到多重共线性的问题。相关性矩阵图可以用漂亮的可视化方式表示相关性。如下所示：

```
> corrplot(water.cor, method = "ellipse")
```

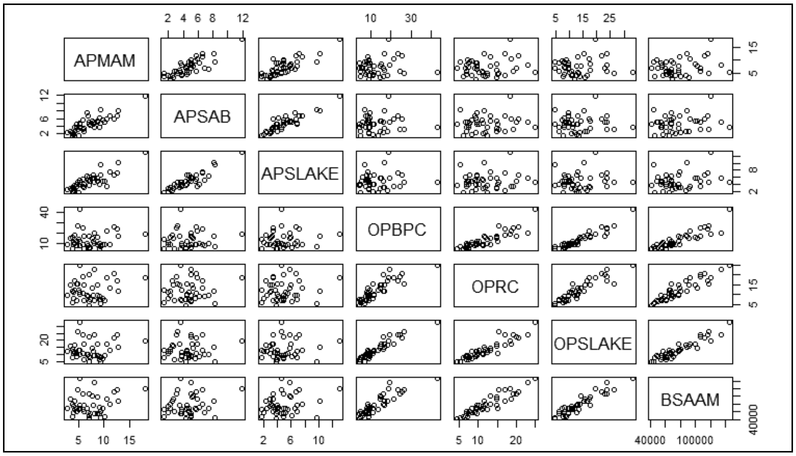
上述代码片段输出如下。



另外一种常用的可视化方式是散点图矩阵，可以通过调用`pairs()`函数来实现。它可以使我们看到比前面的相关性矩阵图更多的信息：

```
> pairs(~ ., data = socal.water)
```

上述代码片段输出如下。



2.2.3　模型构建与模型评价

本节要讨论的一个关键问题就是特征选择，这个任务特别重要。在本章中，我们要讨论最优子集回归和逐步回归方法，使用leaps包。后面的章节会涉及更高级的技术。

前向逐步选择从一个零特征模型开始，然后每次添加一个特征，直到所有特征添加完毕。在这个过程中，被添加的选定特征建立的模型具有最小的RSS。所以理论上，第一个选定的特征应该能最好地解释响应变量，依此类推。

> 添加一个特征一定会使RSS减少，使R方增加，但不一定能提高模型的拟合度和可解释性。

后向逐步回归从一个包含所有特征的模型开始，每次删除一个起最小作用的特征。现在也有一种混合方法，这种算法先通过前向逐步回归添加特征，然后检查是否有特征不再对提高模型拟合度起作用，如果有则删除。每次建模之后，分析者都可以检查模型输出，并使用各种统计量选择能提供最佳拟合的特征。

这里我要给出一个重要提示，逐步回归技术会遇到非常严重的问题。对于一个数据集，你先用前向逐步回归，然后再用后向逐步回归，可能会得到两个完全矛盾的模型。最重要的一点是，逐步回归会使回归系数发生偏离，换句话说，会使回归系数的值过大，置信区间过窄（Tibrishani，1996）。

对于特征选择，最优子集回归是逐步回归的一个可接受的替代方案。在最优子集回归中，算法使用所有可能的特征组合来拟合模型，所以，如果有3个特征，将生成7个模型。然后和逐步回归一样，分析者需要应用自己的判断和统计分析来选择最优模型，模型选择就是后面工作的关键。正如你猜想的那样，如果数据集有多个特征，工作量就会非常大。当特征数多于观测数时（p大于n），这个方法的效果就不会好。

当然，这些对于最优子集法的限制不会影响我们现在要做的工作，基于现有的限制条件，我们放弃逐步回归。当然，你完全可以试试。首先加载leaps包。为了查看特征选择如何进行，我们先用所有特征建立和检查模型，然后逐渐深入，使用最优子集法选择最优的拟合。

要想使用所有特征构建线性模型，仍然可以使用`lm()`函数。模型形式为fit = lm (y~$x_1 + x_2 + x_3 + \cdots + x_n$)。一个简单的技巧是，如果你想包括所有特征，只要在波形符号后面加一个句号，而不用把它们全打出来。对于初学者来说，我们要加载leaps包，建立一个包含所有特征的模型进行研究，如下所示：

```
    > library(leaps)

> fit <- lm(BSAAM ~ ., data = socal.water)

> summary(fit)

Call:
lm(formula = BSAAM ~ ., data = socal.water)

Residuals:
   Min     1Q Median    3Q    Max
-12690  -4936  -1424  4173  18542

Coefficients:
    Estimate Std. Error t value Pr(>|t|)
(Intercept) 15944.67    4099.80   3.889 0.000416
      ***
APMAM         -12.77     708.89  -0.018 0.985725
APSAB        -664.41    1522.89  -0.436 0.665237
APSLAKE      2270.68    1341.29   1.693 0.099112 .
OPBPC          69.70     461.69   0.151 0.880839
OPRC         1916.45     641.36   2.988 0.005031 **
OPSLAKE      2211.58     752.69   2.938 0.005729 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 '' 1
Residual standard error: 7557 on 36 degrees of
      freedom
Multiple R-squared: 0.9248,     Adjusted R-squared:
      0.9123
F-statistic: 73.82 on 6 and 36 DF, p-value: <
      2.2e-16
```

与单变量回归一样，我们要检查F统计量的p值，以检验是否至少有一个非零系数。确实，p值是高度显著的。还可以看到，`OPRC`和`OPSLAKE`这两个参数具有显著的p值。有趣的是，`OPBPC`并不显著，尽管它与响应变量高度相关。简言之，当我们控制其他OP开头的特征时，`OPBPC`无法对预测方差提供任何有意义的解释。这就是说，模型中存在`OPRC`和`OPSLAKE`时，特征`OPBPC`从统计学角度来看没有任何作用。

建立初始模型之后，使用最优子集法。我们使用leaps包中的`regsubsets()`函数建立一个

sub.fit对象。如下所示：

```
> sub.fit <- regsubsets(BSAAM ~ ., data =
   socal.water)
```

这样就生成了best.summary对象，帮助我们更深入地研究模型。对于R中的所有对象，都可以使用names()函数列出输出结果。如下所示：

```
    > best.summary <- summary(sub.fit)

> names(best.summary)
[1] "which"  "rsq"    "rss"   "adjr2"   "cp"
       "bic"    "outmat" "obj"
```

其他对于模型选择有价值的函数还有which.min()和which.max()，它们分别给出具有某个输出的最小值和最大值的模型，如下代码片段所示：

```
    > which.min(best.summary$rss)
[1] 6
```

以上代码告诉我们，有6个特征的模型具有最小的RSS。本应如此，因为它有最多的输入，输入越多，RSS越小。请注意，增加特征必然会减少RSS！而且必然会增加R方。我们可以添加一个完全不相关的特征，比如洛杉矶湖人队的胜场数，RSS也会减少，R方也会增加。变动值可能微不足道，但聊胜于无。看来，我们需要一种切实有效的方法来恰当地选择相关特征。

本章将讨论4种用于特征选择的统计方法：**赤池信息量准则**、**马洛斯的Cp**、**贝叶斯信息量准则**和修正R方。前三种方法的目标是追求统计量的值最小化，修正R方的目标是追求统计量的值最大化。这些统计方法的目的是建立一个尽可能简约的模型，换句话说，要对模型复杂性进行“惩罚”。

上述4种统计量的公式如下。

- $\text{AIC} = n * \log\left(\frac{\text{RSS}_p}{n}\right) + 2 * p$，$p$为被检验模型中的特征数量。
- $\text{CP} = \frac{\text{RSS}_p}{\text{MSE}_f} - n + 2 * p$，$p$为被检验模型中的特征数量，$\text{MSE}_f$是包含所有特征的模型误差的平方均值（均方误差），$n$为样本大小。
- $\text{BIC} = n * \log\left(\frac{\text{RSS}_p}{n}\right) + p * \log(n)$，$p$为被检验模型中的特征数量，$n$为样本大小。
- $\text{修正R方} = 1 - \left(\frac{\text{RSS}}{n - p - 1}\right) / \left(\frac{\text{R方}}{n - 1}\right)$，$p$为被检验模型中的特征数量，$n$为样本大小。

在线性模型中，AIC和Cp成正比，所以我们只需关注Cp，在leaps包的输出中可以找到它。

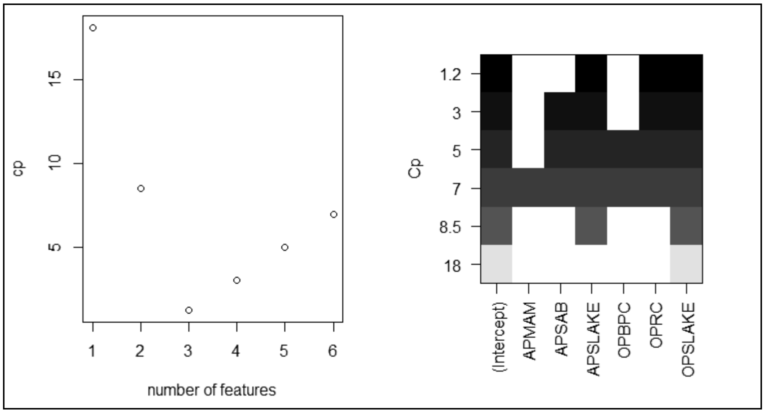
BIC与Cp相比，更倾向于选择变量较少的模型，所以我们要对二者进行比较。为此，生成两张并列的统计图进行分析。使用下面的代码片段比较Cp和BIC：

```
    > par(mfrow = c(1,2))

> plot(best.summary$cp, xlab = "number of
      features", ylab = "cp")

> plot(sub.fit, scale = "Cp")
```

上述代码片段输出如下。



在左侧的图中，可以看出有3个特征的模型具有最小的Cp值。在右侧的图中，显示了能给出最小Cp的特征组合。这张图应该这么看：先在Y轴的最高点找到最小的Cp值，此处是1.2；然后向右在X轴上找到与之对应的色块。通过这张图，我们可以看到这个具有最小Cp值的模型中的3个特征是**APSLAKE**、**OPRC**和**OPSLAKE**。通过`which.min()`和`which.max()`函数，我们可以进行Cp与BIC和修正R方的比较。

```
    > which.min(best.summary$bic)
[1] 3

> which.max(best.summary$adjr2)
[1] 3
```

可以看出，在本例中，BIC、修正R方和Cp选择的最优模型是一致的。现在，与单变量线性回归一样，我们需要检查模型并进行假设检验。正像之前做的那样，我们需要生成一个线性模型对象并检查统计图。如下所示：

```
    > best.fit <- lm(BSAAM ~ APSLAKE + OPRC + OPSLAKE,
      data =
      socal.water)

> summary(best.fit)
```

```
Call:
lm(formula = BSAAM ~ APSLAKE + OPRC + OPSLAKE)

Residuals:
   Min    1Q Median   3Q   Max
-12964 -5140 -1252 4446 18649

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  15424.6     3638.4   4.239 0.000133
    ***
APSLAKE       1712.5      500.5   3.421 0.001475 **
OPRC          1797.5      567.8   3.166 0.002998 **
OPSLAKE       2389.8      447.1   5.346 4.19e-06
    ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1

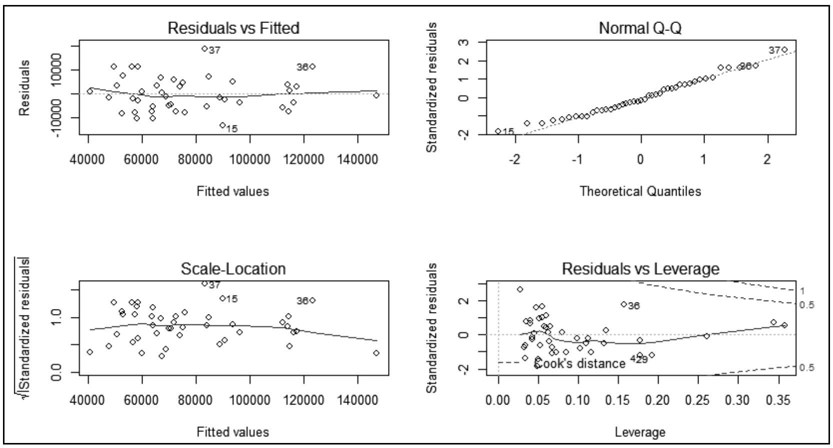
Residual standard error: 7284 on 39 degrees of
      freedom
Multiple R-squared:  0.9244, Adjusted R-squared:
      0.9185
F-statistic: 158.9 on 3 and 39 DF, p-value: <
      2.2e-16
```

在这个三特征模型中，F统计量和所有*t*检验都具有显著的*p*值。通过了第一个检验，我们即可生成诊断图，如下所示：

```
    > par(mfrow = c(2,2))

> plot(best.fit)
```

上述代码片段输出如下。



通过这4张图，我们完全可以认为，残差具有固定的方差并且服从正态分布。杠杆图中也没有什么需要我们进一步处理的异常。

为了研究共线性的问题，我们要引入**方差膨胀因子**这个统计量。VIF是一个比率，分子为使用全部特征拟合模型时该特征的系数的方差，分母为仅使用该特征拟合模型时这个特征的系数的方差。计算公式为$1/(1-R_i^2)$，其中R_i^2为以第i个特征作为响应变量，其余所有特征作为解释变量进行线性回归得到的R方值。VIF能取得的最小值是1，表示根本不存在共线性。现在还没有一个确定的准则决定共线性的严重程度，一般认为，VIF值超过5（有些人认为是10）就说明存在严重的共线性（James，p.101，2013）。我们难以选定一个精确的阈值，因为没有一个确定的统计学标准来决定多重共线性什么时候会使模型变得不可接受。

car包中的`vif()`函数完全可以计算出VIF值，参见下面的代码片段：

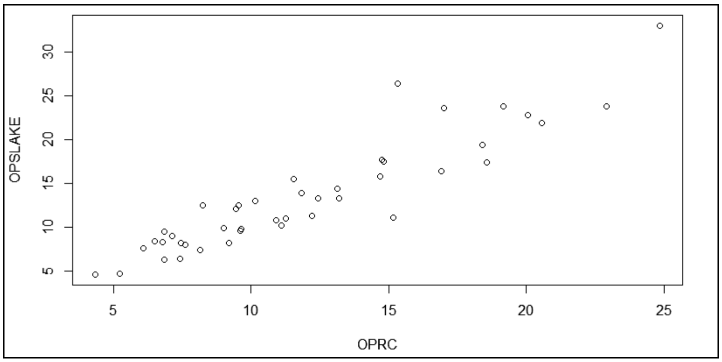
```
    > vif(best.fit)

APSLAKE     OPRC OPSLAKE
1.011499 6.452569 6.444748
```

基于相关性分析，我们发现OPRC和OPSLAKE存在潜在的共线性问题（VIF值大于5），这也没什么可大惊小怪的。这两个变量的关系图揭示了问题的根源，参见下面的屏幕截图。

```
> plot(socal.water$OPRC, socal.water$OPSLAKE, xlab
  = "OPRC", ylab = "OPSLAKE")
```

上述命令输出如下。



解决共线性的简单方式就是，在不影响预测能力的前提下去掉这个变量。看一下最优子集法中生成的修正R方的值就会发现，APSLAKE和OPSLAKE组成的两变量模型的值为0.90，加入OPRC之后仅有一个微不足道的提升，到了0.92。如下所示：

```
    > best.summary$adjr2 #adjusted r-squared values
[1] 0.8777515 0.9001619 0.9185369 0.9168706
      0.9146772 0.9123079
```

看一下这个两变量模型及其假设检验结果，如下所示：

```
> fit.2 <- lm(BSAAM ~ APSLAKE+OPSLAKE, data =
  socal.water)

> summary(fit.2)

Call:
lm(formula = BSAAM ~ APSLAKE + OPSLAKE)

Residuals:
          Min       1Q  Median       3Q     Max
-13335.8  -5893.2   -171.8  4219.5  19500.2

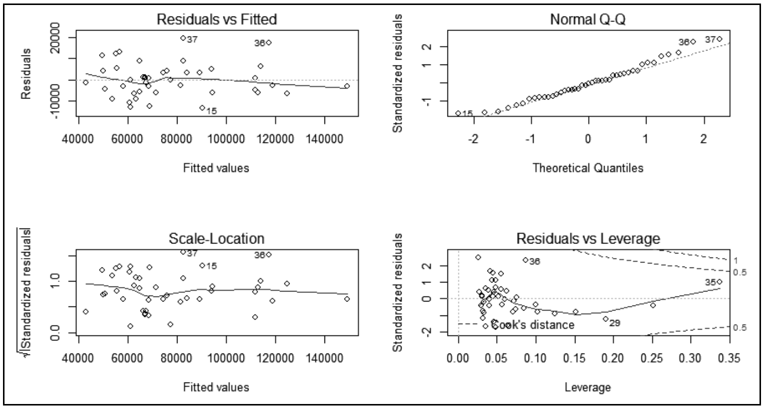
Coefficients:
        Estimate Std. Error t value Pr(>|t|)
(Intercept)  19144.9     3812.0   5.022  1.1e-05
    ***
APSLAKE       1768.8      553.7   3.194  0.00273 **
OPSLAKE       3689.5      196.0  18.829  < 2e-16
    ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1

Residual standard error: 8063 on 40 degrees of
      freedom
Multiple R-squared:  0.9049,    Adjusted R-squared:
      0.9002
F-statistic: 190.3 on 2 and 40 DF,  p-value: <
      2.2e-16

> par(mfrow=c(2,2))

> plot(fit.2)
```

上述代码片段输出如下。



模型是显著的，诊断图中也没发现什么问题，共线性问题应该得到了解决。再使用`vif()`函数检查一下，如下所示：

```
> vif(fit.2)

 APSLAKE  OPSLAKE
1.010221 1.010221
```

如前所述，我不认为残差与拟合图会有什么问题。如果你不相信，可以用R对误差的同方差性进行正式的假设检验。这个检验称为**Breusch-Pagan**（**BP**）检验。要想做这个检验，需要加载lmtest包，然后运行一行代码。BP检验的原假设是误差方差为0，对应的备择假设是误差方差不为0。

```
> library(lmtest)

> bptest(fit.2)

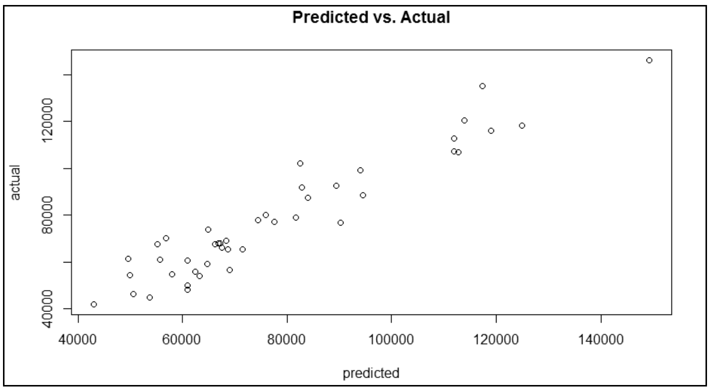
  studentized Breusch-Pagan test
data:  fit.2
BP = 0.0046, df = 2, p-value = 0.9977
```

我们没有证据拒绝认为"误差方差为0"的原假设，因为p值 = 0.9977。检验结果中，BP = 0.0046是一个卡方值。

所有的事情都搞定了，看上去，最好的预测模型应该由APSLAKE和OPSLAKE两个特征组成。这个模型可以解释河川径流量的90%的方差。为预测流量，应该用19 145（截距）加上1769乘以ASLAKE的值，再加上3690乘以OPSLAKE的值。在R基础包中，使用模型拟合值与响应变量的值可以生成**预测值**相对于**实际值**的散点图。如下所示：

```
> plot(fit.2$fitted.values, socal.water$BSAAM, xlab
 = "predicted", ylab = "actual", main = "Predicted
   vs.Actual")
```

上述代码片段输出如下。



尽管很有信息量，但R的基础图形功能依然不适用于商业性展示。但在R中，我们可以轻松美化图形。现在有很多R包加强了图形功能，我们此处将使用ggplot2。生成统计图之前，必须将预测值放入数据框socal.water。还要把BSAAM重命名为Actual，并在数据框中加入一个新的向量，如下所示：

```
    > socal.water["Actual"] = water$BSAAM #create the
      vector Actual

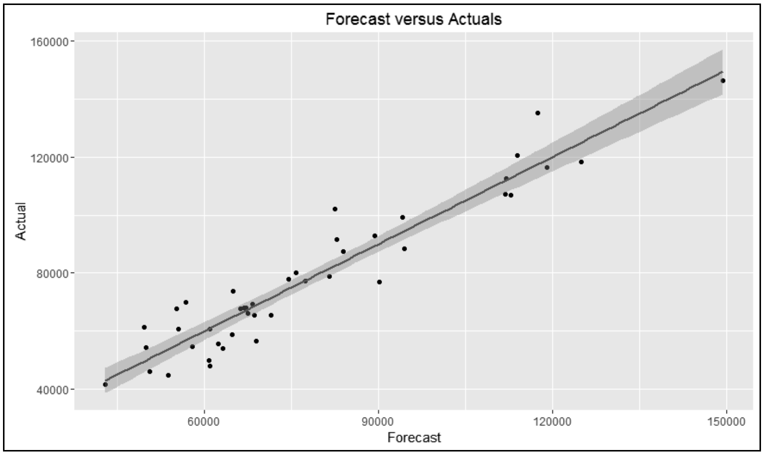
> socal.water$Forecast = predict(fit.2) #populate
     Forecast with the predicted values
```

然后加载ggplot2包，使用一行代码生成一个更漂亮的统计图：

```
    > library(ggplot2)

> ggplot(socal.water, aes(x = Forecast, y =
      Actual)) + geom_point() + geom_smooth(method =
         lm) + labs(title = "Forecast versus Actuals")
```

上述代码片段输出如下。



继续本章内容之前，我们先讨论一项终极的模型选择技术——交叉验证。交叉验证用于模型选择和检验，这种有效方法得到了广泛应用。交叉验证为什么是必须的？这还要归因于偏差/方差的权衡问题。美国莱特州立大学的塔皮教授对此有精彩的论述：

> “我们经常使用回归模型预测未来观测值。我们用数据去拟合模型是完全可以的，但如果用来预测模型响应的数据和用来估计模型的数据属于同一批，那么说预测效果有多么好就有欺骗的嫌疑了。在评价模型对未来观测值的预测效果方面，这样做往往会得到过于乐观的结果。如果我们留下一个观测值，用其余观测值拟合模型，然后再预测这个留下的观测值，那么评价模型预测效果时，得到的结论会具有更少偏差。”

塔皮教授在上述论述中提出的交叉验证技术被称为**留一法交叉验证**。在线性模型中，你可以很容易地进行LOOCV，检测**预测误差平方和**这个统计量，然后选择具有最小值的模型即可。R中的MPV库可以计算这个统计量，代码如下：

```
> library(MPV)

> PRESS(best.fit)
[1] 2426757258

> PRESS(fit.2)
[1] 2992801411
```

如果仅参考这个统计量，我们应该选择模型best.fit。但如前所述，我还是认为更简约的模型更好。你可以利用简洁优雅的矩阵代数，自己编写一个简单的函数来计算这个统计量，代码如下所示：

```
> PRESS.best = sum((resid(best.fit)/(1 -
   hatvalues(best.fit)))^2)

> PRESS.fit.2 = sum((resid(fit.2)/(1 -
   hatvalues(fit.2)))^2)

> PRESS.best
[1] 2426757258

> PRESS.fit.2
[1] 2992801411
```

你可能会问，什么是hatvalues（帽子值）？如果你有一个线性模型$Y = B_0 + B_1x + e$，我们可以将其转换为矩阵表示形式：$Y = XB + E$。在这种表示形式下，Y保持不变，X是输入值矩阵，B是系数，E代表误差。从这个线性模型可以解出B的值。不用进行繁冗的矩阵乘法运算，回归过程可以得到一个所谓**帽子矩阵**。这个矩阵将模型中计算的值映射（或称“投影”）到实际值。因此，它反映了一个特定的观测值在模型中有多大的影响力。所以，先用残差除以1减去帽子值的差，再对结果求平方和，最后就可以得到与LOOCV相同的结果。

2.3　线性模型中的其他问题

进行下一章之前，关于线性模型还有两个额外的问题需要讨论。第一个问题是如何使模型包含定性特征，第二个问题是如何处理交互项。

2.3.1　定性特征

定性特征经常被称为“因子”，它可以有两个或更多个水平，比如“男/女”或“差/中/好”。如果我们有一个具有两个水平的特征，比如性别，那么可以建立一个指标，或称“虚拟特征”。

任意地将一个水平设为0，另一个水平设为1。如果只用这个指标建立模型，那么线性模型的形式还是和前面一样，即$Y = B_0 + B_1x + e$。如果对特征进行编码，使男 = 0，女 = 1，那么当特征为“男”时，模型的期望结果就是截距B_0；当特征为“女”时，模型的期望结果就是$B_0 + B_1x$。如果特征的水平多于两个，你就可以建立$n - 1$个指标；所以3个水平需要两个指标。如果你建立的指标数和水平数相同，就会掉入虚拟变量陷阱，导致完全的多重共线性。

我们通过一个简单的例子学习如何解释输出结果。加载ISLR包，使用Carseats数据集建立一个模型，代码片段如下：

```
    > library(ISLR)

> data(Carseats)

> str(Carseats)

'data.frame':   400 obs. of 11 variables:
$ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
$ CompPrice  : num  138 111 113 117 141 124 115 136
       132 132 ...
$ Income     : num  73 48 35 100 64 113 105 81 110
       113 ...
$ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
$ Population : num  276 260 269 466 340 501 45 425
       108 131 ...
$ Price      : num  120 83 80 97 128 72 108 120 124
       124 ...
$ ShelveLoc : Factor w/ 3 levels
       "Bad","Good","Medium": 1 2 3 3 1
      1 3 2 3 3 ...
$ Age       : num 42 65 59 55 38 78 71 67 76 76
      ...
$ Education : num 17 10 12 14 13 16 15 10 10 17
      ...
$ Urban     : Factor w/ 2 levels "No","Yes": 2 2 2
      2 2 1 2 2 1 1
      ...
$ US        : Factor w/ 2 levels "No","Yes": 2 2 2
      2 1 2 1 2 1 2
      ..
```

在这个例子中，我们将预测Carseats中的sales变量，仅用一个定量特征Advertising和一个定性特征ShelveLoc。定性特征是一个因子，具有3个水平：Bad、Good和Medium。对于因子，R会在分析时自动对指标进行编码。模型的建立和分析如下所示：

```
    > sales.fit <- lm(Sales ~ Advertising + ShelveLoc,
       data = Carseats)

> summary(sales.fit)

Call:
```

```
lm(formula = Sales ~ Advertising + ShelveLoc, data =
Carseats)

Residuals:
    Min      1Q  Median      3Q     Max
-6.6480 -1.6198 -0.0476 1.5308 6.4098

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
(Intercept)      4.89662    0.25207  19.426  < 2e-
      16 ***
Advertising      0.10071    0.01692   5.951 5.88e-
      09 ***
ShelveLocGood    4.57686    0.33479  13.671  < 2e-
      16 ***
ShelveLocMedium  1.75142    0.27475   6.375 5.11e-
      10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1

Residual standard error: 2.244 on 396 degrees of
      freedom
Multiple R-squared:  0.3733,    Adjusted R-squared:
      0.3685
F-statistic: 78.62 on 3 and 396 DF, p-value: <
      2.2e-16
```

截距的值是4.89662，如果货架位置好，销售量的估计值几乎是货架位置差的两倍。如果想知道R如何对指标特征进行编码，可以使用`contrasts()`函数，如下所示：

```
    > contrasts(Carseats$ShelveLoc)

        Good Medium
Bad        0      0
Good       1      0
Medium     0      1
```

2.3.2　交互项

R中对交互项的处理也很容易。当一个特征的预测效果依赖于另一个特征的值时，这两个特征就是交互作用的。有交互项的模型可以表示为$Y = B_0 + B_1x_1 + B_2x_2 + B_1B_2x_1x_2 + e$。MASS包中提供了一个现成的例子，使用了Boston数据集。响应变量为房屋价值的中位数，用medv表示。我们用两个特征作为预测变量，一个是低社会经济地位家庭百分比，用lstat表示；一个是房龄年数，用age表示。代码及输出如下：

```
    > library(MASS)

> data(Boston)
```

```
> str(Boston)

'data.frame':    506 obs. of  14 variables:
$ crim   : num   0.00632 0.02731 0.02729 0.03237
       0.06905 ...
$ zn     : num   18 0 0 0 0 0 12.5 12.5 12.5 12.5
       ...
$ Indus  : num   2.31 7.07 7.07 2.18 2.18 2.18 7.87
       7.87 7.87 7.87
      ...
$ chas   : int   0 0 0 0 0 0 0 0 0 0 ...
$ nox    : num   0.538 0.469 0.469 0.458 0.458 0.458
       0.524 0.524
       0.524 0.524 ...
$ rm     : num   6.58 6.42 7.18 7 7.15 ...
$ age    : num   65.2 78.9 61.1 45.8 54.2 58.7 66.6
       96.1 100 85.9
       ...
$ dis    : num   4.09 4.97 4.97 6.06 6.06 ...
$ rad    : int   1 2 2 3 3 3 5 5 5 5 ...
$ tax    : num   296 242 242 222 222 222 311 311 311
       311 ...
$ ptratio: num   15.3 17.8 17.8 18.7 18.7 18.7 15.2
      15.2 15.2 15.2
      ...
$ black  : num   397 397 393 395 397 ...
$ lstat  : num   4.98 9.14 4.03 2.94 5.33 ...
$ medv   : num   24 21.6 34.7 33.4 36.2 28.7 22.9
      27.1 16.5 18.9 ...
```

在`lm()`函数中使用feature1*feature2，将两个特征及其交互项加入模型，如下所示：

```
> value.fit <- lm(medv ~ lstat * age, data =
  Boston)

> summary(value.fit)

Call:
lm(formula = medv ~ lstat * age, data = Boston)

Residuals:
    Min      1Q  Median      3Q      Max
-15.806 -4.045  -1.333   2.085   27.552

Coefficients:
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.0885359 1.4698355  24.553   < 2e-16
      ***
lstat       -1.3921168  0.1674555  -8.313  8.78e-16
      ***
age -0.0007209 0.0198792 -0.036 0.9711
lstat:age 0.0041560 0.0018518 2.244 0.0252
      *
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1

Residual standard error: 6.149 on 502 degrees of
      freedom
Multiple R-squared: 0.5557, Adjusted R-squared:
      0.5531
F-statistic: 209.3 on 3 and 502 DF, p-value: <
      2.2e-16
```

检查输出可以知道，社会经济地位是个具有高预测性的特征，而房龄则不是。但这两个变量具有显著的交互作用，可以对房屋价值进行正向解释。

2.4　小结

在机器学习的语境下，我们训练和检测模型，用来预测或预报结果。在本章中，我们深入而彻底地研究了线性回归方法，它预测定量结果，虽然简单，但极其有效。后续章节会涉及更高级的技术，但其中很多技术只是本章所学内容的扩展。我们还讨论了不对数据集进行可视化检查，而只依赖统计量进行模型选择时可能出现的问题。

通过寥寥几行代码，你就可以做出非常强大的有洞察力的预测，以支持决策。这不仅简单有效，还可以在特征中包含定量变量和交互项。在机器学习领域内深耕的每个人都绝对应该精通线性回归。

第 3 章

逻辑斯蒂回归与判别分析

"我们这个世界的真正逻辑寓于概率的计算之中。"

——詹姆斯·克拉克·麦克斯韦，苏格兰物理学家

我们在第2章学习了如何使用**普通最小二乘法**预测定量结果，换句话说，就是线性回归。是时候换个主题了，下面将学习如何使用算法预测定性结果。这样的结果变量可能是二值变量（如男/女、是否购买、良性/恶性肿瘤），也可能是多值分类（如教育水平、眼睛颜色）。不管这种结果变量是二值的还是多值的，分析的任务都是对一个观测属于结果变量的某个类别的概率做出预测。换句话说，我们开发算法的目的是对观测进行分类。

开始研究分类问题之前，我们先讨论为什么OLS线性回归不是解决分类问题的合适技术，以及本章将要介绍的算法是如何解决这类问题的。然后讨论一个具体问题，预测肿瘤活体检查的结果是良性还是恶性。我们使用的数据集非常著名，这个数据集通常被称为"威斯康星乳腺癌数据集"。为解决这个问题，首先建立一个逻辑斯蒂回归模型并进行解释，然后研究模型检测方法以选择特征，期望得到最合适的模型。接下来，讨论线性判别分析方法和二次判别分析方法，并将这两种方法与逻辑斯蒂回归进行比较。在此之后，使用乳腺癌数据建立预测模型。最后介绍多元样条回归方法，以及选择解决实际问题的最佳综合算法，从而圆满解决整个问题。这些方法将为后续章节中的更高级机器学习方法打下基础。

3.1 分类方法与线性回归

既然我们在前一章学习了最小二乘回归方法，为什么不能把它用于定性结果呢？事实证明，你可以用，但后果自负。不妨假设你有一个结果需要预测，结果有三类：轻微、中等、严重。你和你的同事认为轻微与中等之间的差别，以及中等与严重之间的差别是相等的，是一个线性关系。你可以建立虚拟变量，用0表示轻微，1表示中等，2表示严重。如果你有理由相信这个，那么线性回归就是一个可接受的解决方案。但是，定性估计（如前面的例子）有可能产生高测量误差，从而使OLS发生偏离。多数商业问题没有一个在科学上能够被接受的方法，可以将一个定性的响应变量转化为定量的响应变量。如果我们有一个双结果响应变量，比如考试通过和考试失败，应

该怎么做？可以使用虚拟变量，将“考试失败”编码为0，“考试通过”编码为1，然后使用线性回归建立一个模型，模型预测值为观测考试通过还是考试失败的概率。但是概率范围为[0, 1]，模型中Y的估计值非常有可能超出这个范围。如果这样，那就有点不好解释了。

3.2　逻辑斯蒂回归

如前所述，分类问题最好使用位于0和1之间的概率进行建模。对于所有观测，我们有很多不同的函数可以实现这个功能，本章重点讨论逻辑斯蒂函数。逻辑斯蒂回归中使用的逻辑斯蒂函数如下所示：

$$Y\text{的概率} = e^{B_0+B_1} / 1 + e^{B_0+B_1x}$$

如果你曾经在赛马比赛或世界杯中下过一些小小的赌注，那就可以通过赔率更好地理解这个概念。逻辑斯蒂函数可以通过公式Probability(Y)/(1 − Probability(Y))转化为赔率。举例来说，如果巴西队赢得世界杯的概率是20%，那么赔率就是0.2/(1 − 0.2)，等于0.25，用赔率的话讲就是“1赔4”。

要从赔率回到概率，就要用赔率除以1加赔率的和。在世界杯的例子中，就是0.25/(1 + 0.25)，等于20%。此外，我们再考虑一下优势比。假设德国队赢得世界杯的赔率是0.18，我们可以通过优势比来比较巴西队和德国队的赔率。在本例中，优势比为巴西队的赔率除以德国队的赔率。可以得到优势比为0.25/0.18，等于1.39。这样，我们可以说巴西队赢得世界杯的可能性是德国队的1.39倍。

有一种方法可以看出逻辑斯蒂回归和线性回归之间的联系，逻辑斯蒂回归就是以对数发生比为响应变量进行线性拟合，即log(P(Y)/1 − P(Y)) =B_0+B_1x。这里的系数是通过极大似然估计得到的，而不是通过OLS。极大似然的直观意义就是，我们要找到一对B_0和B_1的估计值，使它们产生的对观测的预测概率尽可能接近Y的实际观测结果，这就是所谓的似然性。与其他软件一样，R语言也可以实现极大似然估计，通过找到最优的B值组合来使似然性最大化。

请记住，逻辑斯蒂回归是一项强大的技术，可以预测分类问题，通常也是对分类问题进行建模的起点。因此，对于本章中将要面临的商业问题，我们使用逻辑斯蒂回归作为解决问题的首选方法。

3.2.1　业务理解

威斯康星大学的威廉·沃尔伯格博士1990年发布了威斯康星乳腺癌数据集。他收集数据的目标是想辨别肿瘤活体检查结果是良性的还是恶性的。他的团队使用**细针穿刺**（FNA）技术收集样本。如果医生通过检查发现肿瘤或者通过透视发现异常组织，下一步就是进行活体检查。FNA是取得活体组织的安全方法，很少出现并发症。病理学家对活体组织进行检查，试图确定诊断结果

（恶性或良性）。正如你所想像的，诊断结果并非无足轻重。良性的乳腺肿瘤并不危险，因为不存在异常组织扩散到身体其他部分的风险。如果良性肿瘤过大，就需要通过外科手术切除。从另一方面来说，恶性肿瘤必须进行医疗干预，治疗的程度与很多因素有关，一般都要进行外科手术，然后进行放射性治疗或化学治疗。所以，误诊造成的后果是很严重的。误诊为恶性（false positive，假阳性）会导致昂贵但不必要的治疗费用，还会使患者背负巨大的心理和生理负担。另一方面，误诊为良性（false negative，假阴性）会使患者得不到应有的治疗，造成癌细胞扩散，引起过早死亡。乳腺癌患者的早期医疗干预可以大大提高存活率。

我们的任务是开发尽可能准确的机器学习诊断算法，帮助医疗团队确定肿瘤性质。

3.2.2　数据理解和数据准备

数据集包含699个患者的组织样本，保存在有11个变量的数据框中，如下所示。

- `ID`：样本编码
- `V1`：细胞浓度
- `V2`：细胞大小均匀度
- `V3`：细胞形状均匀度
- `V4`：边缘黏着度
- `V5`：单上皮细胞大小
- `V6`：裸细胞核（16个观测值缺失）
- `V7`：平和染色质
- `V8`：正常核仁
- `V9`：有丝分裂状态
- `class`：肿瘤诊断结果，良性或恶性；这就是我们要预测的结果变量。

医疗团队对9个特征进行了评分和编码，评分的范围是1~10。

可以在R的MASS包中找到该数据框，名为biopsy。为进行数据准备，我们加载这个数据框，确认数据结构，将变量重命名为有意义的名称，还要删除有缺失项的观测，然后即可开始对数据进行可视化探索。下面就是我们开始工作的代码，首先加载库文件和数据集，然后使用`str()`函数检查数据内部结构。如下所示：

```
    > library(MASS)
> data(biopsy)

> str(biopsy)
'data.frame':   699 obs. of   11 variables:
$ ID    : chr  "1000025" "1002945" "1015425"
       "1016277" ...
$ V1    : int  5 5 3 6 4 8 1 2 2 4 ...
```

```
$ V2    : int  1 4 1 8 1 10 1 1 1 2 ...
$ V3    : int  1 4 1 8 1 10 1 2 1 1 ...
$ V4    : int  1 5 1 1 3 8 1 1 1 1 ...
$ V5    : int  2 7 2 3 2 7 2 2 2 2 ...
$ V6    : int  1 10 2 4 1 10 10 1 1 1 ...
$ V7    : int  3 3 3 3 3 9 3 3 1 2 ...
$ V8    : int  1 2 1 7 1 7 1 1 1 1 ...
$ V9    : int  1 1 1 1 1 1 1 1 5 1 ...
$ class: Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1
    1 1 ...
```

对数据结构的检查发现，特征是整数型变量，结果变量是一个因子，不需要将数据转换为其他结构。

我们可以删除ID列，如下所示：

```
> biopsy$ID = NULL
```

接着重命名变量，并确认操作结果：

```
> names(biopsy) <- c("thick", "u.size", "u.shape",
   "adhsn", "s.size", "nucl", "chrom", "n.nuc",
       "mit", "class")
> names(biopsy)
[1] "thick"   "u.size" "u.shape" "adhsn"
   "s.size"  "nucl"
   "chrom"   "n.nuc"
[9] "mit"     "class"
```

现在，我们要删除那些有缺失数据的观测。既然只有16个观测有缺失数据，所以删除它们不会有什么危险，因为它们只占所有观测的2%。我们不会在本章对如何处理缺失数据做全面讨论，你可以参考附录A，我在那里介绍了如何对数据进行处理。通过删除这些观测，我们建立了一个新的工作数据框。使用na.omit()函数后，一行代码就可以巧妙删除所有有缺失数据的观测。

```
> biopsy.v2 <- na.omit(biopsy)
```

根据你分析数据时使用的R包的不同，结果变量也可能要求是数值型，比如0或者1。为了满足这个要求，可以创建一个变量y，用0表示良性，用1表示恶性，然后使用ifelse()函数为y赋值，如下所示：

```
> y <- ifelse(biopsy$class == "malignant", 1, 0)
```

在分类问题中，我们可以通过很多方式来可视化地理解数据，我认为用什么方式都是个人的选择。在本例中，我喜欢做的一件事是检查各个特征的箱线图，并将这些箱线图按照分类结果并列排列。如果想知道哪个特征对于算法可能是重要的，那么这是一种特别好的方法，可以一目了然地看出数据的分布特点。根据我的经验，箱线图还提供了一种有效的故事表达方式，你可以直接展示给客户。有很多方法可以快速画出箱线图，lattice包和ggplot2包都可以非常好地完成这个任务。在本例中，我会使用ggplot2包以及一个扩展包reshape2。加载这些程序包之后，你需要用

melt()函数建立一个数据框，这样做是为了在特征融合后就建立一个箱线图矩阵，使我们可以轻松地进行下面的可视化检查：

```
> library(reshape2)
> library(ggplot2)
```

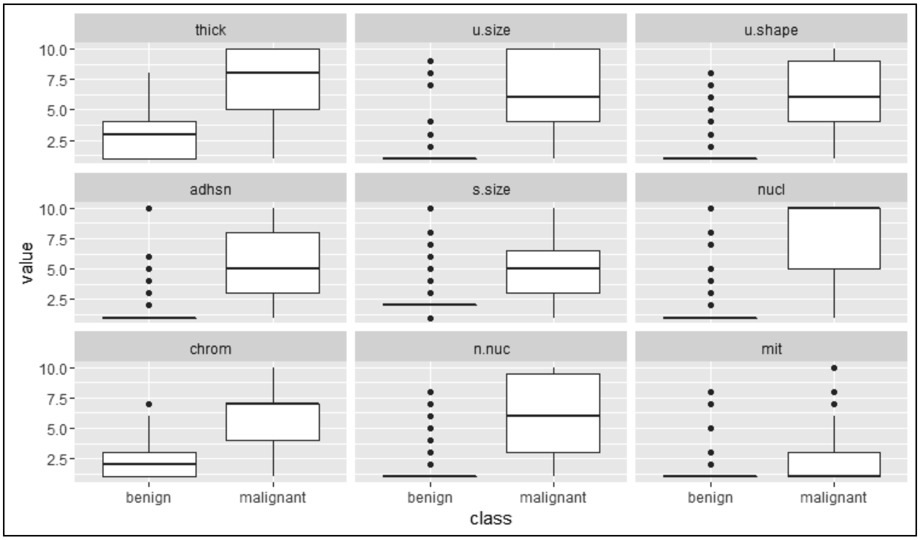
下面的代码将数据按值融合成一个总体特征，并按照class变量进行分组：

```
> biop.m <- melt(biopsy.v2, id.var = "class")
```

通过功能强大的ggplot2，我们可以建立一个3×3的箱线图矩阵，如下所示：

```
> ggplot(data = biop.m, aes(x = class, y = value))
+ geom_boxplot() + facet_wrap(~ variable, ncol = 3)
```

上述代码输出如下。



我们怎么解释箱线图呢？首先，在上面的屏幕截图中，白色矩形表示数据的上四分位数和下四分位数。换句话说，有一半的观测落在白色矩形范围内。白色矩形中间的黑色粗线段表示中位数。从矩形延伸出的直线与四分位距有关，如果没有离群点，则会终止于数据的最大值和最小值。黑点表示离群点。

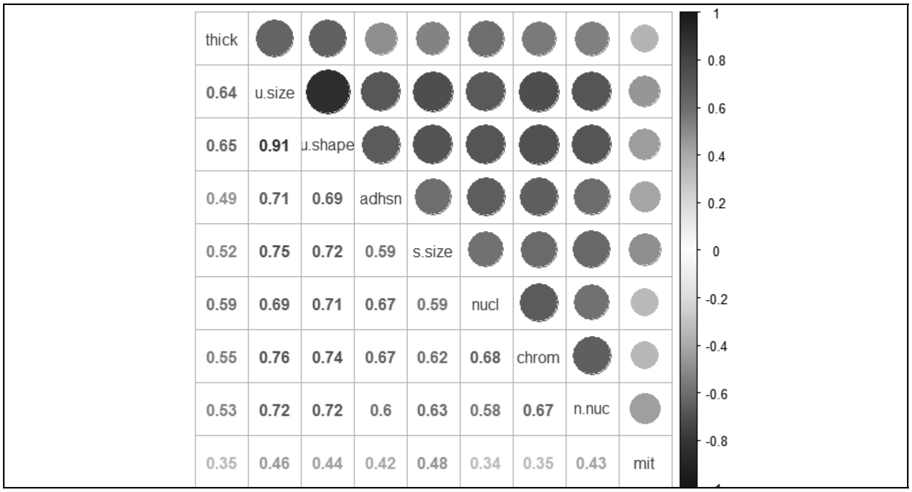
通过对上面箱线图的研究和判断，我们发现，很难确定哪个特征在分类算法中是重要的。但是，从中位数的间隔距离和相关分布来看，我觉得可以有把握地认为nuclei是一个重要特征。与之相反，不同class组的mitosis特征的中位数几乎没有区别，这说明它很可能是个无关特征。下面我们就来看一下！

既然所有特征都是定量的，我们就可以像线性回归一样进行相关性分析。正如我们在线性回

归中讨论过的，逻辑斯蒂回归中的共线性也会使估计发生偏离。加载corrplot包，检查特征之间的相关性，就像在前一章中做的那样。这次使用另一种类型的相关性矩阵图，其中既有渐变椭圆，也有相关系数。如下所示：

```
> library(corrplot)
> bc <- cor(biopsy.v2[, 1:9]) #create an object of
      the features
> corrplot.mixed(bc)
```

上述代码输出如下。



从相关系数可以看出，我们会遇到共线性问题，特别是细胞大小均匀度和细胞形状均匀度表现出非常明显的共线性。作为逻辑斯蒂回归建模过程的一部分，VIF分析是必不可少的，正如我们在线性回归中所做的那样。数据准备的最终任务是建立训练数据集和测试数据集。从原始数据中建立两个不同的数据集的目的在于，这样可以更准确地预测未使用数据或未知数据。

实际上，在机器学习中，我们不应该过于关心对已使用的观测的预测有多好，而应该把重点放在那些建立算法时没有使用过的观测的预测上面。所以，我们要使用训练数据建立并选择一个对测试数据能做出最好预测的最优算法。在本章中，我们会遵循这个评价标准来建立模型。

有多种方式可以将数据恰当地划分成训练集和测试集：50/50、60/40、70/30、80/20，诸如此类。你应该根据自己的经验和判断选择数据划分方式。在本例中，我选择按照70/30的比例划分数据。如下所示：

```
> set.seed(123) #random number generator
> ind <- sample(2, nrow(biopsy.v2), replace = TRUE,
      prob = c(0.7, 0.3))
> train <- biopsy.v2[ind==1, ] #the training data
      set
```

```
> test <- biopsy.v2[ind==2, ] #the test data set
> str(test) #confirm it worked
'data.frame':   209 obs. of  10 variables:
 $ thick  : int  5 6 4 2 1 7 6 7 1 3 ...
 $ u.size : int  4 8 1 1 1 4 1 3 1 2 ...
 $ u.shape: int  4 8 1 2 1 6 1 2 1 1 ...
 $ adhsn  : int  5 1 3 1 1 4 1 10 1 1 ...
 $ s.size : int  7 3 2 2 1 6 2 5 2 1 ...
 $ nucl   : int  10 4 1 1 1 1 1 10 1 1 ...
 $ chrom  : int  3 3 3 3 3 4 3 5 3 2 ...
 $ n.nuc  : int  2 7 1 1 1 3 1 4 1 1 ...
 $ mit    : int  1 1 1 1 1 1 1 4 1 1 ...
 $ class  : Factor w/ 2 levels benign","malignant":
       1 1 1 1 1 2 1
       2 1 1 ...
```


为了确保两个数据集的结果变量是均衡的，我们做以下检查：

```
    > table(train$class)
   benign malignant
     302       172
> table(test$class)
   benign malignant
      142       67
```

两个数据集的结果变量的内部比例完全可以接受，既然如此，下面开始进行模型构建和模型评价。

3.2.3 模型构建与模型评价

在流程的这一部分，我们首先用所有输入变量建立一个逻辑斯蒂回归模型，然后用最优子集法对特征进行削减。在此之后，我们会试验**判别分析**和**多元自适应回归样条**（**MARS**）方法。

1. 逻辑斯蒂回归模型

我们已经讨论了逻辑斯蒂回归背后的理论，现在可以开始拟合模型了。R中的`glm()`函数可以拟合广义线性模型，这是一系列模型，其中包括逻辑斯蒂回归模型。代码的语法和前一章中用过的`lm()`函数很相似，其中一个较大区别是我们必须在函数中使用`family = binomial`这个参数。该参数告诉R运行逻辑斯蒂回归，而不是其他版本的广义线性模型。首先在训练数据集上使用所有特征建立一个模型，看看这个模型在测试数据集上运行的效果。如下所示：

```
    > full.fit <- glm(class ~ ., family = binomial,
      data = train)
> summary(full.fit)
Call:
glm(formula = class ~ ., family = binomial, data =
      train)
Deviance Residuals:
```

```
    Min       1Q    Median      3Q       Max
-3.3397  -0.1387   -0.0716  0.0321    2.3559
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -9.4293     1.2273  -7.683 1.55e-14
      ***
thick         0.5252     0.1601   3.280 0.001039 **
u.size       -0.1045     0.2446  -0.427 0.669165
u.shape       0.2798     0.2526   1.108 0.268044
adhsn         0.3086     0.1738   1.776 0.075722 .
s.size        0.2866     0.2074   1.382 0.167021
nucl          0.4057     0.1213   3.344 0.000826
      ***
chrom         0.2737     0.2174   1.259 0.208006
n.nuc         0.2244     0.1373   1.635 0.102126
mit           0.4296     0.3393   1.266 0.205402
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to
       be 1)
    Null deviance: 620.989  on 473  degrees of
       freedom
Residual deviance:  78.373  on 464  degrees of
       freedom
AIC: 98.373
Number of Fisher Scoring iterations: 8
```

通过`summary()`函数，我们可以查看各个预测变量的系数及其p值。可以看到，只有两个特征的p值小于0.05（thickness和nuclei）。使用`confint()`函数可以对模型进行95%置信区间的检验。如下所示：

```
    > confint(full.fit)
                   2.5 %     97.5 %
(Intercept) -12.23786660 -7.3421509
thick         0.23250518  0.8712407
u.size       -0.56108960  0.4212527
u.shape      -0.24551513  0.7725505
adhsn        -0.02257952  0.6760586
s.size       -0.11769714  0.7024139
nucl          0.17687420  0.6582354
chrom        -0.13992177  0.7232904
n.nuc        -0.03813490  0.5110293
mit          -0.14099177  1.0142786
```

请注意，两个显著特征的置信区间不包括0。对于逻辑斯蒂回归模型的系数，你不能解释为“当X改变1个单位时Y会改变多少”。这时，优势比的作用就体现出来了。对数函数$\log(P(Y)/1-P(Y)) = B_0 + B_1x$的$B$系数可以通过`exponent(beta)`指数函数转化为优势比。

要在R中计算优势比，可以使用下面的`exp(coef())`函数形式：

```
> exp(coef(full.fit))
 (Intercept)        thick        u.size      u.shape
     adhsn
8.033466e-05 1.690879e+00 9.007478e-01 1.322844e+00
     1.361533e+00
s.size         nucl         chrom        n.nuc  mit
1.331940e+00 1.500309e+00 1.314783e+00 1.251551e+00
    1.536709e+00
```


优势比可以解释为特征中1个单位的变化导致的结果发生比的变化。如果系数大于1，就说明当特征的值增加时，结果的发生比会增加。反之，系数小于1就说明，当特征的值增加时，结果的发生比会减小。在本例中，除u.size之外的所有特征都会增加对数发生比。

我们进行数据探索时发现一个问题——潜在的多重共线性。同线性回归一样，在逻辑斯蒂回归中也可以算出VIF统计量，如下所示：

```
> library(car)
> vif(full.fit)
   thick u.size u.shape adhsn s.size nucl
      chrom  n.nuc
1.2352 3.2488 2.8303 1.3021 1.6356 1.3729
      1.5234 1.3431
  mit
1.059707
```

没有一个VIF值大于5，根据VIF经验法则，共线性看来不成为一个问题。下面的任务就是进行特征选择，先别急，我们先编写一些代码，看看模型在训练数据集和测试数据集上表现如何。

首先要建立一个向量，表示预测概率，如下所示：

```
> train.probs <- predict(full.fit, type =
    "response")
> train.probs[1:5] #inspect the first 5 predicted
        probabilities
[1] 0.02052820 0.01087838 0.99992668 0.08987453
        0.01379266
```

下一步需要评价模型在训练集上执行的效果，然后再评价它在测试集上的拟合程度。快速实现评价的方法是生成一个混淆矩阵。在后面的章节中，我们使用的混淆矩阵是由caret包实现的，InformationValue包也可以实现混淆矩阵。这时，我们需要用0和1来表示结果。函数区别良性结果和恶性结果使用的默认值是0.50，也就是说，当概率大于等于0.50时，就认为这个结果是恶性的：

```
> trainY <- y[ind==1]
> testY <- y[ind==2]
> confusionMatrix(trainY, train.probs)
    0    1
0 294    7
1   8  165
```

矩阵的行表示预测值，列表示实际值。对角线上的元素是预测正确的分类。右上角的值7表示

误为恶性的预测数，左下角的值8表示误为良性的预测数。可以查看错误预测的比例。如下所示：

```
> misClassError(trainY, train.probs)
[1] 0.0316
```

看上去我们干得相当不错，训练集上只有3.16%的预测错误率。如前所述，我们必须正确预测未知数据，换句话说，必须正确预测测试集。

在测试集上建立混淆矩阵的方法和在训练集上一样：

```
> test.probs <- predict(full.fit, newdata = test,
   type = "response")
> misClassError(testY, test.probs)
[1] 0.0239
> confusionMatrix(testY, test.probs)
    0    1
0 139    2
1   3   65
```

看上去，我们使用全部特征建立的模型效果非常好，差不多98%的预测正确率真是让人激动。但是，我们还是要看看是否有改进的空间。想象一下，假如你或你的亲人是被误诊的患者，会怎么样？就像前面提到的，后果很严重。出于这种考虑，还可能有更好的方式来建立分类算法吗？下面我们就来看一下！

2. 使用交叉验证的逻辑斯蒂回归

交叉验证的目的是提高测试集上的预测正确率，以及尽可能避免过拟合。**K折交叉验证**的做法是将数据集分成K个相等的等份，每个等份称为一个**K子集**（K-set）。算法每次留出一个子集，使用其余$K-1$个子集拟合模型，然后用模型在留出的那个子集上做预测。将上面K次验证的结果进行平均，可以使误差最小化，并且获得合适的特征选择。你也可以使用**留一交叉验证**方法，这里的K等于N。模拟表明，LOOCV可以获得近乎无偏的估计，但是会有很高的方差。所以，大多数机器学习专家都建议将K的值定为5或10。

bestglm包可以自动进行交叉验证，这个包依赖于我们在线性回归中使用过的leaps包。交叉验证的语法和数据格式存在注意事项，所以我们按部就班地进行：

```
    > library(bestglm)
Loading required package: leaps
```

加载程序包之后，需要将结果编码成0或1。如果结果仍为因子，程序将不起作用。使用这个程序包的另外一个要求是结果变量（或y）必须是最后一列，而且要删除所有没有用的列。通过以下代码新建一个数据框即可快速实现上述要求：

```
> X <- train[, 1:9]
> Xy <- data.frame(cbind(X, trainY))
```

以下代码使用数据进行交叉验证：

```
> bestglm(Xy = Xy, IC="CV",
  CVArgs=list(Method="HTF", K=10,
  REP=1), family=binomial)
```

在上面的代码中，`Xy = Xy`指的是我们已经格式化的数据框，`IC = "CV"`告诉程序使用的信息准则为交叉验证，`CVArgs`是我们要使用的交叉验证参数。`HTF`方法就是K折交叉验证，后面的数字`K = 10`指定了均分的份数，参数`REP = 1`告诉程序随机使用等份并且只迭代一次。像`glm()`函数一样，我们需要指定参数`family = binomial`。顺便说一句，如果指定参数`family = gaussian`，你就可以使用`bestglm()`函数进行线性回归。完成上面的交叉验证分析之后，会得到下面的结果：Best Model有三个特征，它们是`thick`、`u.size`和`nucl`。Morgan-Tatar搜索的结果表明，程序对所有可能的子集进行了简单而彻底的搜索。如下所示：

```
    Morgan-Tatar search since family is non-gaussian.
CV(K = 10, REP = 1)
BICq equivalent for q in (7.16797006619085e-05,
    0.273173435514231)
Best Model:
Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -7.8147191 0.90996494 -8.587934
     8.854687e-18
thick         0.6188466 0.14713075  4.206100
     2.598159e-05
u.size        0.6582015 0.15295415  4.303260
     1.683031e-05
nucl          0.5725902 0.09922549  5.770596
     7.899178e-09
```

我们可以把这些特征放到`glm()`函数中，看看模型在测试集上表现如何。`predict()`函数不能用于bestglm生成的模型，所以下面的步骤是必需的：

```
> reduce.fit <- glm(class ~ thick + u.size + nucl,
   family = binomial, data = train)
```

同前面一样，下面的代码可以在测试集上比较预测值和实际值：

```
> test.cv.probs <- predict(reduce.fit, newdata =
   test, type = "response")
> misClassError(testY, test.cv.probs)
[1] 0.0383
> confusionMatrix(testY, test.cv.probs)
    0    1
0 139    5
1   3   62
```

精简了特征的模型和全特征模型相比，精确度略有下降，但这并不是“世界末日”。我们可以用bestglm包再试一次，这次使用信息准则为BIC的最优子集方法：

```
    > bestglm(Xy = Xy, IC = "BIC", family = binomial)
Morgan-Tatar search since family is non-gaussian.
BIC
```

```
BICq equivalent for q in (0.273173435514231,
      0.577036596263757)
Best Model:
 Estimate Std. Error    z value     Pr(>|z|)
(Intercept) -8.6169613  1.03155250 -8.353391
      6.633065e-17
thick         0.7113613  0.14751510  4.822295
      1.419160e-06
adhsn         0.4537948  0.15034294  3.018398
      2.541153e-03
nucl          0.5579922  0.09848156  5.665956
      1.462068e-08
n.nuc         0.4290854  0.11845720  3.622282
      2.920152e-04
```

试过所有可能的子集之后，以上4个特征提供了最小的BIC评分。我们看看这个模型在测试集上的预测效果，如下所示：

```
    > bic.fit <- glm(class ~ thick + adhsn + nucl +
        n.nuc, family = binomial, data = train)
> test.bic.probs <- predict(bic.fit, newdata =
        test, type = "response")
    > misClassError(testY, test.bic.probs)
    [1] 0.0239
    > confusionMatrix(testY, test.bic.probs)
        0    1
    0 138    1
    1   4   66
```

我们得到5个错误的预测，和全特征模型一样。那么问题来了：哪一个模型更好？在任何正常情况下，如果具有相同的泛化效果，经验法则会选择最简单的或解释性最好的模型。我们当然可以开启一个全新的分析，使用新的随机数以及新的训练集和测试集划分比例。但是，不妨假定我们已经做了逻辑斯蒂回归能做的所有操作，最后还是回到了全特征模型和BIC最小模型，仍然需要讨论模型选择的方法。下面讨论判别分析的方法，在最后的模型推荐中，我们有可能选择使用这种方法建立的模型。

3.3　判别分析概述

判别分析又称**费舍尔判别分析**，也是一项常用的分类技术。当分类很确定时，判别分析可以有效替代逻辑斯蒂回归。当你遇到分类结果很确定的分类问题时，逻辑斯蒂回归的估计结果可能是不稳定的，即置信区间很宽，不同样本之间的估计值会有很大变化（James，2013）。判别分析不会受到这个问题的困扰，实际上，它会比逻辑回归做得更好，泛化能力更强。反之，如果特征和结果变量之间具有错综复杂的关系，判别分析在分类任务上的表现就会非常差。在乳腺癌这个例子中，逻辑斯蒂回归在训练集和测试集上的表现都非常好，分类的结果并不确定。出于同逻辑回归进行比较的目的，我们研究一下判别分析，包括**线性判别分析**和**二次判别分析**。

判别分析使用**贝叶斯定理**确定每个观测属于某个类别的概率。如果你有两个类别，比如良性和恶性，判别分析会计算观测分别属于两个类别的概率，然后选择高概率的类别作为正确的类别。

贝叶斯定理定义了在X已经发生的条件下Y发生的概率——等于Y和X同时发生的概率除以X发生的概率，公式如下：

$$Y / X\text{的概率} = \frac{\mathrm{P}(X+Y)}{\mathrm{P}(X)}$$

公式中的分子表示一个具有某些特征的观测属于某个分类水平的可能性，分母表示一个具有这些特征的观测属于所有分类水平的可能性。同样地，分类原则认为如果X和Y的联合分布已知，那么给定X后，决定观测属于哪个类别的最佳决策是选择那个有更大概率（后验概率）的类别。

获得后验概率的过程如下所示。

(1) 收集已知类别的数据。

(2) 计算先验概率——代表属于某个类别的样本的比例。

(3) 按类别计算每个特征的均值。

(4) 计算每个特征的方差–协方差矩阵。在线性判别分析中，这会是一个所有类别的混合矩阵，给出线性分类器；在二次判别分析中，会对每个分类建立一个方差–协方差矩阵。

(5) 估计每个分类的正态分布（高斯密度）。

(6) 计算`discriminant`函数，作为一个新对象的分类原则。

(7) 根据`discriminant`函数，将观测分配到某个分类。

我们还可以给出确定后验概率的扩展表示形式，如下所示。

- ❑ π_k = 分类k中的样本数/总体样本数，是一个随机选择的观测属于第k个分类的先验概率。
- ❑ $f_k(X) = \mathrm{P}(X = x \mid Y = k)$ 是一个观测属于第k个分类的概率密度函数。假设概率服从正态分布（高斯分布）。如果有多个特征，假设概率服从多元高斯分布。
- ❑ 通过$p_k(X)$ = Y在给定X时的概率，我们按以下方式调整贝叶斯定理。$-p_x(X) = \pi_k f_k(X) / \sum_{l=1}^{k} \pi_l f_l(X)$。这是给定一个观测的特征值时，这个观测属于第$k$个分类的后验概率。
- ❑ 假设$k = 2$并且先验概率相同，$\pi_1 = \pi_2$，则当$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$时，观测值被分配到第一个分类中，否则就被分配到第二个分类。这个公式称为决策边界。判别分析会生成$k-1$个决策边界，也就是说，如果有三个类别（$k = 3$），那么就会有两个决策边界。

尽管线性判别分析简单而又优雅，它仍然具有局限性。线性判别分析假设每种类别中的观测服从多元正态分布，并且不同类别之间的具有同样的协方差。二次判别分析仍然假设观测服从正态分布，但假设每种类别都有自己的协方差。

为什么这种假设很重要？当你放宽相同协方差假设，就意味着允许二次项进入判别分数的计

算，这在线性判别分析中是不可能的。其中的数学意义理解起来有些挑战性，已经超出了本书的讨论范围。重要的是要记住，二次判别分析技术比逻辑斯蒂回归更具灵活性，同时还要时刻牢记偏差—方差权衡的问题。使用更有灵活性的技术可以得到偏差更小的结果，但很可能具有更高的方差。和很多灵活的技术一样，需要一个高鲁棒性的训练数据集来降低高分类方差。

判别分析应用

线性判别分析（LDA）可以用MASS包实现，我们为了使用biopsy数据集已经加载了这个包。LDA的语法和`lm()`以及`glm()`函数非常相似。

现在可以开始LDA模型的拟合了，如下所示：

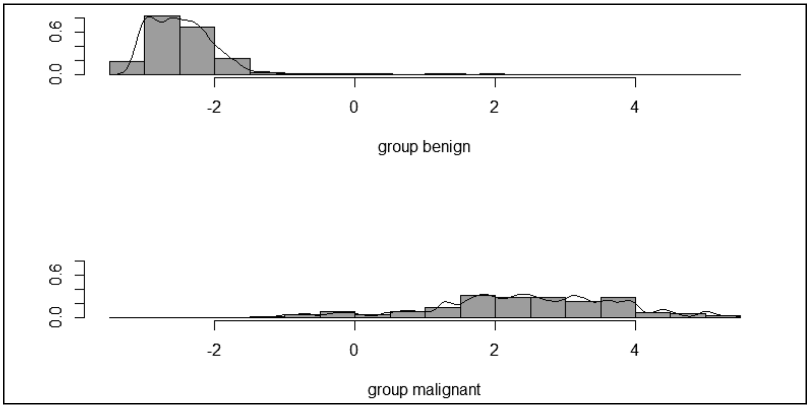
```
    > lda.fit <- lda(class ~ ., data = train)
> lda.fit
Call:
lda(class ~ ., data = train)
Prior probabilities of groups:
   benign malignant
0.6371308 0.3628692
Group means:
    thick  u.size u.shape   adhsn  s.size    nucl
      chrom
benign    2.9205 1.30463 1.41390 1.32450 2.11589
      1.39735 2.08278
malignant 7.1918 6.69767 6.68604 5.66860 5.50000
      7.67441 5.95930
            n.nuc     mit
benign    1.22516 1.09271
malignant 5.90697 2.63953
Coefficients of linear discriminants:
                LD1
Thick    0.19557291
u.size   0.10555201
u.shape  0.06327200
adhsn    0.04752757
s.size   0.10678521
nucl     0.26196145
chrom    0.08102965
n.nuc    0.11691054
mit     -0.01665454
```

从结果可以看出，在分组先验概率中，良性概率大约为64%，恶性概率大约为36%。下面再看看分组均值，这是按类别分组的每个特征的均值。线性判别系数是标准线性组合，用来确定观测的判别评分的特征。评分越高，越可能被分入恶性组。

对LDA模型使用`plot()`函数，可以画出判别评分的直方图和密度图，如下所示：

```
> plot(lda.fit, type = "both")
```

上述命令输出如下。



可以看出，组间有些重合，这表明有些观测被错误分类。

LDA模型可以用`predict()`函数得到3种元素（class、posterior和x）的列表。class元素是对良性或恶性的预测，posterior是值为x的评分可能属于某个类别的概率，x是线性判别评分。通过下面的函数，我们仅提取恶性观测的概率：

```
    > train.lda.probs <- predict(lda.fit)$posterior[,
      2]
> misClassError(trainY, train.lda.probs)
    [1] 0.0401
    > confusionMatrix(trainY, train.lda.probs)
        0    1
    0 296   13
    1   6  159
```

很不幸，我们的LDA模型在训练集上表现得比逻辑斯蒂回归模型差多了。但最重要的是LDA模型在测试集上的表现，如下所示：

```
> test.lda.probs <- predict(lda.fit, newdata =
   test)$posterior[, 2]
> misClassError(testY, test.lda.probs)
[1] 0.0383
> confusionMatrix(testY, test.lda.probs)
    0    1
0 140    6
1   2   61
```

基于LDA模型在训练集上的糟糕表现，它在测试集上的表现比我预期的要好。从正确分类率的角度看，LDA模型表现得依然不如逻辑斯蒂回归模型（LDA模型：96%，逻辑斯蒂回归模型：98%）。

下面用二次判别分析（QDA）模型来拟合数据。在R中，QDA也是MASS包的一部分，函数为`qda()`。建模过程直截了当，我们将模型存储在一个名为`qda.fit`的对象中，如下页所示：

```
    > qda.fit = qda(class ~ ., data = train)
> qda.fit
Call:
qda(class ~ ., data = train)
Prior probabilities of groups:
   benign malignant
0.6371308 0.3628692
Group means:
    Thick u.size u.shape adhsn s.size   nucl  chrom
     n.nuc
benign    2.9205 1.3046 1.4139 1.3245 2.1158
      1.3973 2.0827 1.2251
malignant 7.1918 6.6976 6.6860 5.6686 5.5000
      7.6744 5.9593 5.9069
               mit
benign    1.092715
malignant 2.639535
```

结果中有分组均值，和LDA一样；但是没有系数，因为这是二次函数，我们前面讨论过了。

使用与LDA相同的代码，可以得到QDA模型在训练集和测试集上的预测结果：

```
> train.qda.probs <- predict(qda.fit)$posterior[,
  2]
> misClassError(trainY, train.qda.probs)
[1] 0.0422
> confusionMatrix(trainY, train.qda.probs)
    0    1
0 287    5
1  15  167
> test.qda.probs <- predict(qda.fit, newdata =
  test)$posterior[, 2]
> misClassError(testY, test.qda.probs)
[1] 0.0526
> confusionMatrix(testY, test.qda.probs)
    0    1
0 132    1
1  10   66
```

根据混淆矩阵可以立即断定，QDA模型在训练数据集上表现得最差。QDA在测试集上的分类效果也很差，有11个预测错误，而误为恶性的比例尤其高。

3.4　多元自适应回归样条方法

如果有一种建模技术具有以下特点，你是不是会喜出望外？

- 对于回归和分类问题，都可以灵活建立线性模型和非线性模型。
- 支持变量的交互项。
- 简单易懂，易于解释。

- ❑ 几乎不需要数据预处理。
- ❑ 可以处理所有类型的数据：数值型、因子等。
- ❑ 可以很好地预测未知数据，也就是说，在偏差–方差的权衡方面做得非常好。

如果这些特点都让你心动，那么就根本不用我再向你推荐MARS模型了。几个月前我注意到了这种方法，而且发现这种方法的效果出奇地好。实际上，在我最近的工作中，这种方法在测试数据上的效果超过了随机森林和提升树。它立刻就成为了我的基准模型，其他模型都是用来和它做比较的。这种方法的另外一个优点是，它否定了我正在做的很多特征工程方面的工作。这些工作中，很多都要使用**证据权重法**（**WOE**）和**信息价值法**（**IV**）来捕获非线性并对变量进行重新编码。我本来是想花些篇幅介绍WOE和IV方法的，但经过一些测试，我发现这些技术能做的事情，MARS也做得非常好（比如捕获非线性），所以干脆就不再介绍WOE和IV方法了。

要理解MARS非常容易。首先，我们从一个前面已经讨论过的线性模型或广义线性模型开始。然后，为了捕获非线性关系，添加一个铰链（hinge）函数。这些铰链作用于输入特征，相当于系数的改变。举例来说，假设有这样一个模型：Y=12.5（截距）+1.5（变量1）+3.3（变量2），其中变量1和变量2的范围都在1和10之间。下面，我们看看变量2的铰链函数如何发挥作用：

$$Y = 11（新截距）+1.5（变量1）+ 4.26734(\max(0, 变量2 - 5.5))$$

于是，我们可以这样理解铰链函数：在0和变量2减去5.50的结果中，它找出其中的最大值。所以，只要变量2的值大于5.5，铰链函数的值就会乘以系数，否则它的值就是0。这种方法可以给每个变量配置多个铰链，也可以用于交互项。

MARS方法的另一个有趣之处是，它可以自动进行变量选择。这是通过交叉验证实现的，默认的方法是通过前向过程建模——这非常类似于前向逐步回归——然后通过后向过程精简模型。因为模型完成前向过程之后，有可能出现对数据的过拟合，所以要通过后向过程根据**广义交叉验证**对输入特征进行精简，并去除铰链。

$$GCV=RSS/N*(1 - 参数有效数量/N)^2$$
$$参数有效数量 = 输入特征数量 + 惩罚系数*(输入特征数量-1)/2$$

在earth包中，对于加法模型来说，惩罚系数 = 2；对于乘法模型，其中有交互项，惩罚系数 = 3。

在R中，你可以调整的参数非常少。在下面的例子中，我会演示一种简单而又有效实现MARS的方法。如果你的需求很多，那么可以学习Stephen Milborrow制作的非常棒的在线教程Notes on the earth package，通过以下链接可以学习MARS的更多灵活用法：

http://www.milbo.org/doc/earth-notes.pdf

介绍完MARS以后，我们开始学习其用法。你可以使用MDA包，但因为我是通过earth包学习的，所以我还是使用这个包。代码和前面的例子很相似，在那里我们使用了`glm()`函数。但需要

注意，一定要设定如何精简模型，并且使响应变量服从二元分布。我的设定是使用5折交叉验证来选择模型（pmethod="cv"，nfold = 5），重复3次（ncross = 3）；使用没有交互项的加法模型（degree = 1），每个输入特征只使用一个铰链函数（minspan = -1）。在我使用的数据中，交互项和多个铰链函数都会导致过拟合。当然，你的情况会有所不同。代码如下所示：

```
> library(earth)
> set.seed(1)
> earth.fit <- earth(class ~ ., data = train,
                    pmethod = "cv",
                    nfold = 5,
                    ncross = 3,
                    degree = 1,
                    minspan = -1,
                    glm=list(family=binomial)
                    )
```

下面查看模型摘要。乍一看，它有点让人摸不着头脑。当然，我们可以看到模型公式和逻辑斯蒂回归系数，还有铰链函数，再后面是一些注释和广义R方的数值，等等。MARS模型的生成过程是：先根据数据建立一个标准线性回归模型，它的响应变量在内部编码为0和1；对特征（变量）进行精简之后，生成最终模型，并将其转换为一个GLM模型。所以，我们可以不用考虑R方的值：

```
> summary(earth.fit)
Call: earth(formula=class~., data=train,
  pmethod="cv",
glm=list(family=binomial), degree=1, ncross=3,
  nfold=5, minspan=-1)
GLM coefficients
             malignant
(Intercept) -6.5746417
u.size 0.1502747
adhsn 0.3058496
s.size 0.3188098
nucl 0.4426061
n.nuc 0.2307595
h(thick-3) 0.7019053
h(3-chrom) -0.6927319
Earth selected 8 of 10 terms, and 7 of 9 predictors
  using pmethod="cv"
Termination condition: RSq changed by less than
  0.001 at 10 terms
Importance: nucl, u.size, thick, n.nuc, chrom,
  s.size, adhsn,
  u.shape-unused, ...
Number of terms at each degree of interaction: 1 7
  (additive model)
Earth GRSq 0.8354593 RSq 0.8450554 mean.oof.RSq
  0.8331308 (sd 0.0295)
GLM null.deviance 620.9885 (473 dof) deviance
  81.90976 (466 dof)
  iters 8
```

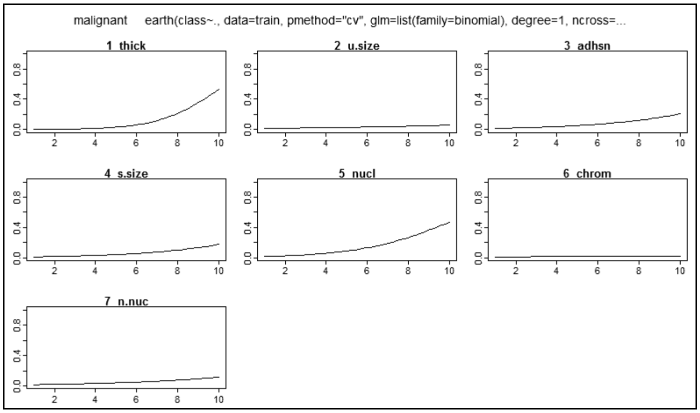
```
pmethod="backward" would have selected the same
  model:
8 terms 7 preds, GRSq 0.8354593 RSq 0.8450554
  mean.oof.RSq
  0.8331308
```

模型有8项，包括截距和7个预测变量。其中两个预测变量有铰链函数，这就是浓度和染色质变量。如果浓度大于3，就会用系数0.7019乘以铰链函数的值，否则这一项就是0。对于染色质，如果它的值小于3，那么就用系数乘以铰链函数值，否则这一项就是0。

还可以做出统计图。第一张图是用plotmo()函数生成的，它展示了保持其他预测变量不变，某个预测变量发生变化时，响应变量发生的改变。你可以清楚地看到铰链函数对浓度所起的作用：

```
> plotmo(earth.fit)
```

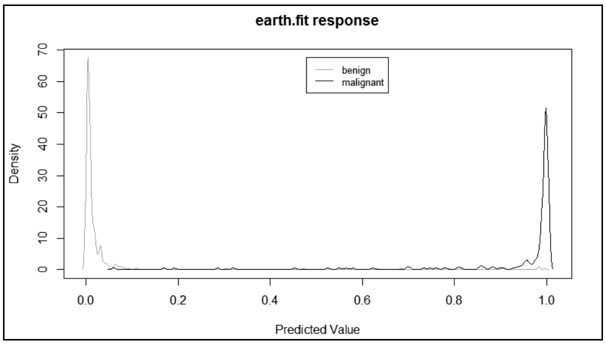
上述命令输出如下。



通过plotd()函数，可以生成按类别标签分类的预测概率密度图：

```
> plotd(earth.fit)
```

上述命令输出如下。



下面看看变量之间的相对重要性。先解释一下，nsubsets是精简过程完成之后包含这个变量的模型的个数。对于gcv和rss列，其中的值表示这个变量贡献的gcv和rss值的减少量（gcv和rss的范围都是0~100）：

```
> evimp(earth.fit)
       nsubsets   gcv    rss
nucl          7 100.0  100.0
u.size        6  44.2   44.8
thick         5  23.8   25.1
n.nuc         4  15.1   16.8
chrom         3   8.3   10.7
s.size        2   6.0    8.1
adhsn         1   2.3    4.6
```

我们再看一下模型在测试集上的表现：

```
> test.earth.probs <- predict(earth.fit, newdata =
  test, type = "response")
> misClassError(testY, test.earth.probs)
[1] 0.0287
> confusionMatrix(testY, test.earth.probs)
    0    1
0 138    2
1   4   65
```

这完全可以和逻辑斯蒂回归模型相比。下面比较前面讲过的各种模型，看看哪个模型是最好的选择。

3.5　模型选择

从上述的工作中，你能得到什么结论？我们从模型中计算出混淆矩阵和错误率，为的就是有一个依据，可是在选择分类模型时还是毫无头绪。对于分类模型的比较，**受试者工作特征**（**ROC**）图是一个很有效的工具。简言之，ROC基于分类器的性能对其进行可视化、组织和选择（Fawcett，2006）。在ROC图中，Y轴是**真阳性率**（**TPR**），X轴是**假阳性率**（**FPR**）。计算过程非常简单，如下所示。

TPR = 正确分类的阳性样本数/所有阳性样本数

FPR = 错误分类的阴性样本数/所有阴性样本数

如果将ROC画出来，可以生成一条曲线，然后可以算出**曲线下面积**了。AUC就是一项衡量分类器性能的有效指标，直观地说，AUC等于一个观测者在面对一对随机选出的案例（其中一个是阳性案例，一个是阴性案例）时，能正确识别出阳性案例的概率（Hanley JA，McNeil BJ，1982）。在我们的例子中，只要将观测者换成我们的算法即可依此进行评价。

要在R中画出ROC图，你可以使用ROCR包。我认为这是一个功能强大的包，它可以让你只

用3行代码就可以画出ROC图。这个程序包还配有一个网站，上面有示例和演示，你可以通过以下网址找到该网站：

http://rocr.bioinf.mpi-sb.mpg.de

下面我要展示4个不同的ROC图：全特征模型、使用BIC选择特征的简化模型、MARS模型和一个糟糕模型。这个所谓的糟糕模型只有1个预测特征，会和其他两个模型形成有效对比。因此，下面加载ROCR包，使用thick这个特征建立这个性能糟糕的模型。为简单起见，我们将其命名为bad.fit。如下所示：


```
  > library(ROCR)
> bad.fit <- glm(class ~ thick, family = binomial,
    data = test)
> test.bad.probs = predict(bad.fit, type =
    "response") #save
    probabilities
```

现在即可使用测试数据集绘制ROC图，每个模型3行代码。首先建立一个对象，保存对实际分类的预测概率，然后使用这个对象建立一个带有TPR和FPR的对象。在此之后，使用`plot()`函数绘制ROC图。我们从全特征模型（我喜欢称之为全模型）开始，它就是我们在3.2节建立的那个初始模型：


```
> pred.full <- prediction(test.probs, test$class)
```

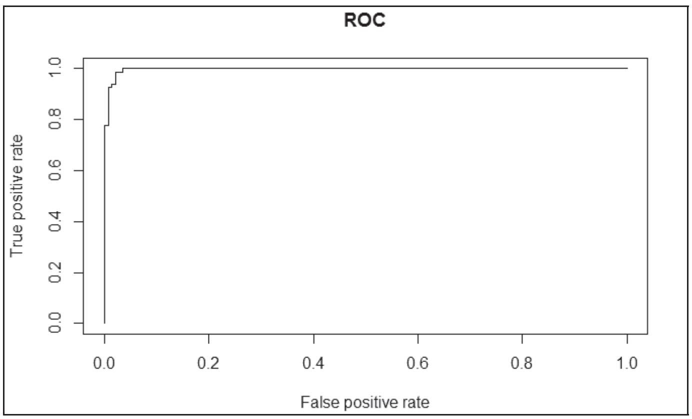
下面是带有TPR和FPR的performance对象：

```
> perf.full <- performance(pred.full, "tpr", "fpr")
```

下面使用`plot()`命令画图，图的标题设为`ROC`，参数`col = 1`指定曲线颜色为黑色：

```
> plot(perf.full, main = "ROC", col = 1)
```

上述命令输入如下。



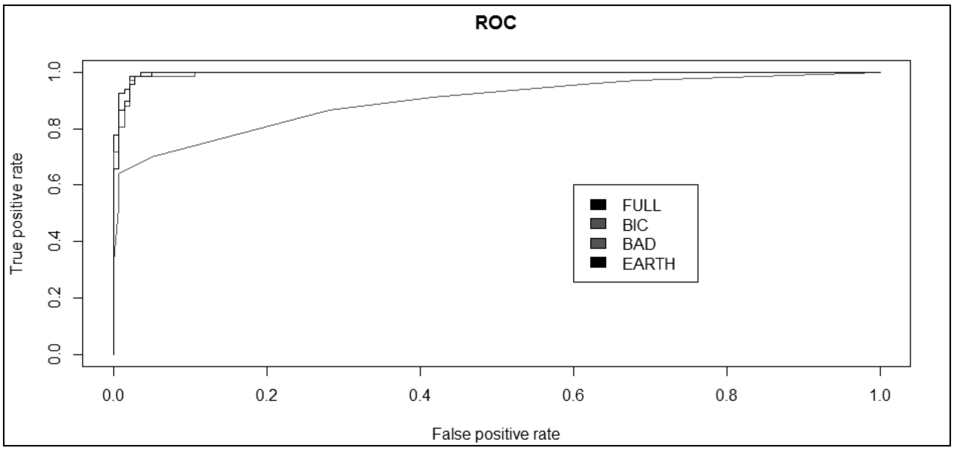
如前所述，曲线的Y轴表示TPR，X轴表示FPR。如果你有一个完美无缺的、没有假阳性错误的分类器，那么就会有一条从X轴0.0处垂直上升的线。如果模型的预测和随机选择没什么区别，那么就会有一条从左下角到右上角的对角线。回忆一下，全模型预测错了5个样本：3个假阳性和2个假阴性。为了便于比较，可以使用同样的代码加上其他模型。首先加上使用BIC建立的模型（参考3.2节），如下所示：

```
    > pred.bic <- prediction(test.bic.probs,
      test$class)
> perf.bic <- performance(pred.bic, "tpr", "fpr")
> plot(perf.bic, col = 2, add = TRUE)
```

`plot()`命令中的`add = TURE`将曲线加入现有图中。最后，加入那个表现糟糕的模型和MARS模型，并加入一个图例，如下所示：

```
    > pred.bad <- prediction(test.bad.probs,
      test$class)
> perf.bad <- performance(pred.bad, "tpr", "fpr")
> plot(perf.bad, col = 3, add = TRUE)
    > pred.earth <- prediction(test.earth.probs,
      test$class)
    > perf.earth <- performance(pred.earth, "tpr",
      "fpr")
    > plot(perf.earth, col = 4, add = TRUE)
    > legend(0.6, 0.6, c("FULL", "BIC", "BAD",
      "EARTH"), 1:4)
```

上述代码片段输出如下。



可以看到，全特征模型、BIC模型和MARS模型基本上重叠在一起。显而易见，**糟糕的**模型表现得和我们预想的一样差。

现在，我们能做的最后一件事就是计算AUC。这也可以通过ROCR包建立performance对象来

实现，只要将tpr和fpr换成auc即可。代码和输出如下所示：

```
> performance(pred.full, "auc")@y.values
[[1]]
[1] 0.9972672
> performance(pred.bic, "auc")@y.values
[[1]]
[1] 0.9944293
> performance(pred.bad, "auc")@y.values
[[1]]
[1] 0.8962056
> performance(pred.earth, "auc")@y.values
[[1]]
[1] 0.9952701
```


最高的AUC值是全模型的99.7%，BIC模型是99.4%，糟糕模型是89.6%，MARS模型是99.5%。所以，从各个方面来看，排除掉糟糕模型，其他几个模型在预测能力方面没有什么区别。我们该怎么做？一个简单的解决方案就是，将训练集和测试集重新随机化，把各种分析再做一遍，比如使用60/40的划分比例和一个不同的随机数种子。但是，如果我们还是得到相同的结果，又该怎么办？我认为，一个纯粹的统计学家会建议选择最简约的模型，但其他人可能更倾向于全特征模型。这就又归结到各种因素的权衡问题了，比如模型准确性与解释性、简约性与扩展性之间的权衡。在本章的例子中，我们完全可以选择更简单的模型，它具有和全模型一样的正确性。不用说你也能知道，仅通过GLM或判别分析不可能总有这样的预测效果。在接下来的章节中，我们会继续研究这个问题，使用一些更复杂的技术来提高模型的预测能力。机器学习的美妙之处就在于——“条条大路通罗马”。

3.6 小结

我们在本章研究了如何使用基于概率的线性模型预测定性响应变量，介绍了三种方法：逻辑斯蒂回归、判别分析和MARS。除此之外，还介绍了ROC图，这是一种可视化的有统计意义的模型选择技术。我们还简要讨论了需要注意的模型选择问题和权衡问题。在后续的章节中，我们还会使用乳腺癌数据集，看看更加复杂的技术在这个数据集上的应用效果。

第4章 线性模型中的高级特征选择技术

“我发现数学变得过于抽象，已经不合我意了，而计算机科学变得过于关心细枝末节——试图在一次计算中存储几毫秒和几千字节。在使用统计学解决实际问题的过程中，我发现了数学和计算机科学的结合之美。”

以上引自斯坦福大学罗伯特·提布施瓦尼教授，参见http://statweb.stanford.edu/~tibs/research_page.html。

迄今为止，我们研究了使用线性模型预测定量和定性结果的方法，着重讨论了特征选择技术，也就是如何排除无用的或不需要的预测变量。我们看到，线性模型在机器学习问题中是非常有效的。但在过去的二十多年中，人们已经开发和提炼了更新的技术，它们提供的预测能力和解释性已经远远超过了我们在前面章节中讨论过的线性模型。当今很多数据集具有数量庞大的特征，即使与观测值的数量相比也毫不逊色，这正如人们所称——高维性。如果你曾经参与过基因组的研究，那么这个问题就不证自明了。此外，随着我们要处理的数据规模不断增大，最优子集和逐步特征选择这样的技术会造成难以承受的时间成本——即使使用高速计算机。我说的可不是以分钟计的时间，在很多情况下，要得到一个最优子集的解需要花费数小时。

这种情况下有更好的解决方式。本章会讨论正则化的概念，正则化会对系数进行限制，甚至将其缩减到0。现在有很多种方法和方法组合可以实现正则化，我们会集中讨论**岭回归**和**最小化绝对收缩和选择算子**，最后讨论**弹性网络**，它将前面两种算法的优点合二为一。

4.1 正则化简介

你应该还记得我们的线性模型形式为$Y = B_0 + B_1x_1 + \cdots + B_nx_n + e$，还有最佳拟合试图最小化RSS。RSS是实际值减去估计值的差的平方和，可以表示为$e_1^2+e_2^2+\cdots+e_n^2$。

通过正则化，我们会在RSS的最小化过程中加入一个新项，称之为**收缩惩罚**。这个惩罚项包

含了一个希腊字母λ以及对β系数和权重的规范化结果。不同的技术对权重的规范化方法都不尽相同，我们随后都要进行相应的讨论。简言之，我们的模型中需要最小化，也就是RSS + λ（规范化后的系数）。我们会对λ进行选择，在模型构建过程中，λ被称为调优参数。请记住，如果$\lambda = 0$，模型就等价于OLS，因为规范化项目都被抵消了。

那么，正则化对我们来说意味着什么？正则化为什么会起作用？首先，正则化方法在计算上非常有效。如果使用最优子集法，我们需要在一个大数据集上测试2^p个模型，这肯定是不可行的。如果使用正则化方法，对于每个λ值，我们只需拟合一个模型，因此效率会有极大提升。另一个理由是我们在前言讨论过的偏差–方差权衡问题。在线性模型中，响应变量和预测变量之间的关系接近于线性，最小二乘估计接近于无偏，但可能有很高的方差。这意味着，训练集中的微小变动会导致最小二乘系数估计结果的巨大变动（James，2013）。正则化通过恰当地选择λ和规范化，可以使偏差–方差权衡达到最优，从而提高模型拟合的效果。最后，系数的正则化还可以用来解决多重共线性的问题。

4

4.1.1 岭回归

我们先研究什么是岭回归，以及它可以做什么和不能做什么。在岭回归中，规范化项是所有系数的平方和，称为**L2-norm**（**L2范数**）。在我们的模型中就是试图最小化RSS + $\lambda(\text{sum}\beta_j^2)$。当$\lambda$增加时，系数会缩小，趋向于0但永远不会为0。岭回归的优点是可以提高预测准确度，但因为它不能使任何一个特征的系数为0，所以在模型解释性上会有些问题，你需要多费一些唇舌。为了解决这个问题，我们使用LASSO。

4.1.2 LASSO

区别于岭回归中的L2-norm，LASSO使用L1-norm，即所有特征权重的绝对值之和，也就是要最小化RSS + $\lambda(\text{sum}|\beta_j|)$。这个收缩惩罚项确实可以使特征权重收缩到0，相对于岭回归，这是一个明显的优势，因为可以极大地提高模型的解释性。

L1-norm为什么能够使权重（或者系数）变为0？其中的数学解释已经超过了本书范围（Tibsharini，1996）。

如果LASSO这么好，那还要岭回归做什么？话别说得这么急！存在高共线性或高度两两相关的情况下，LASSO可能会将某个预测特征强制删除，这会损失模型的预测能力。举例来说，如果特征A和B都应该存在于模型之中，那么LASSO可能会将其中一个的系数缩减到0。下面的引言非常好地总结了这个问题：

> "如果较少数目的预测变量有实际系数，其余预测变量的系数要么非常小，要么为0，那么在这样的情况下，LASSO性能更好。当响应变量是很多预测变量的函数，而且预测

变量的系数大小都差不多时，岭回归表现得更好。”

——James，2013

还是存在两全其美的机会的，这就引出了我们的下一个主题：弹性网络。

4.1.3　弹性网络

弹性网络的强大之处在于，它既能做到岭回归不能做的特征提取，也能实现LASSO不能做的特征分组。重申一下，LASSO倾向于在一组相关的特征中选择一个，忽略其他。弹性网络包含了一个混合参数α，它和λ同时起作用。α是一个0和1之间的数，λ和前面一样，用来调节惩罚项的大小。请注意，当α等于0时，弹性网络等价于岭回归；当α等于1时，弹性网络等价于LASSO。实质上，我们通过对β系数的二次项引入一个第二调优参数，将L1惩罚项和L2惩罚项混合在一起。通过最小化（$\mathrm{RSS} + \lambda[(1 - \alpha)(\mathrm{sum}|\beta_j|^2)/2 + \alpha(\mathrm{sum}|\beta_j|)]/N$）完成目标。

下面我们就来试试这些技术。使用leaps、glmnet和caret包在下面的商业案例中选择合适的特征并生成合适的模型。

4.2　商业案例

本章依然致力于癌症数据的研究——此处案例是前列腺癌。虽然这个数据集比较小，只有97个观测和9个变量，但通过与传统技术的比较，完全可以使你掌握正则化技术的发展。

4.2.1　业务理解

斯坦福大学医疗中心提供了97个病人的术前**前列腺特异性抗原**（PSA）数据，这些病人将要接受彻底的前列腺切除术（切除全部前列腺），以治疗前列腺癌。**美国癌症学会**估计，2014年有30 000名美国男性死于前列腺癌（http://www.cancer.org）。PSA是前列腺分泌的一种蛋白质，发现于前列腺癌患者的血液。我们的目标是，通过临床检测提供的数据建立一个预测模型。对于患者在手术后能够或应该恢复到什么程度，与其他指标相比，PSA可能是一个更有效的预后指标。手术之后，医生会在各个时间区间检查患者的PSA水平，并通过各种公式确定患者是否康复。术前预测模型和术后数据（这里没有提供）互相配合，就可能提高前列腺癌护理的水平，造福于每年数以千计的患者。

4.2.2　数据理解和数据准备

收集自97位男性的数据集保存在一个具有10个变量的数据框中，如下所示。

- lcavol：肿瘤体积的对数值
- lweight：前列腺重量的对数值
- age：患者年龄（以年计）
- lbph：**良性前列腺增生**（BPH）量的对数值，非癌症性质的前列腺增生。
- svi：贮精囊侵入，一个指标变量，表示癌细胞是否已经透过前列腺壁侵入贮精囊（1=是，0=否）。
- lcp：包膜穿透度的对数值，表示癌细胞扩散到前列腺包膜之外的程度。
- gleason：患者的Gleason评分；由病理学家进行活体检查后给出（2~10），表示癌细胞的变异程度——评分越高，程度越危险。
- pgg45：Gleason评分为4或5所占的百分比（高等级癌症）。
- lpsa：PSA值的对数值，响应变量。
- train：一个逻辑向量（TRUE或FALSE，用来区分训练数据和测试数据）。

这个数据集包含在ElemStatLearn这个R包内。加载所需的程序包和数据框之后，查看变量以及变量之间可能存在的联系，如下所示：

```
> library(ElemStatLearn) #contains the data
> library(car) #package to calculate Variance Inflation Factor
> library(corrplot) #correlation plots
> library(leaps) #best subsets regression
> library(glmnet) #allows ridge regression, LASSO and elastic net
> library(caret) #parameter tuning
```

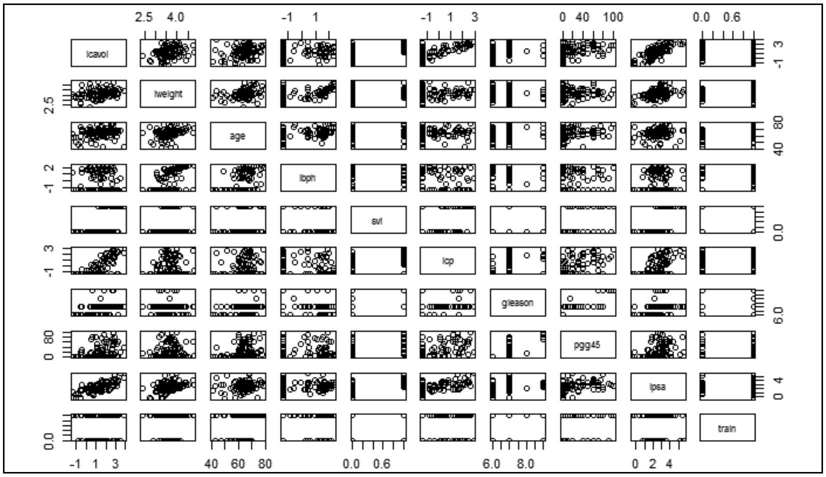
加载程序包之后，调出prostate数据集，查看数据结构，如下所示：

```
> data(prostate)
> str(prostate)
'data.frame':97 obs. of 10 variables:
$ lcavol  : num -0.58 -0.994 -0.511 -1.204 0.751 ...
$ lweight : num 2.77 3.32 2.69 3.28 3.43 ...
$ age     : int 50 58 74 58 62 50 64 58 47 63 ...
$ lbph    : num -1.39 -1.39 -1.39 -1.39 -1.39 ...
$ svi     : int 0 0 0 0 0 0 0 0 0 0 ...
$ lcp     : num -1.39 -1.39 -1.39 -1.39 -1.39 ...
$ gleason : int 6 6 7 6 6 6 6 6 6 6 ...
$ pgg45   : int 0 0 20 0 0 0 0 0 0 0 ...
$ lpsa    : num -0.431 -0.163 -0.163 -0.163 0.372 ...
$ train   : logi TRUE TRUE TRUE TRUE TRUE TRUE ...
```

检查数据结构会发现几个问题，需要特别注意。检查数据集的特征就会发现，svi、lcp、gleason和pgg45的前10个观测值都具有相同的数值，只有一个例外——gleason的第三个观测值。为了保证这些特征作为输入特征是确实可行的，我们可以做出统计图或表格来理解数据。使用下面的plot()命令，将整个数据框作为输入值，即可建立散点图矩阵，如下所示：

```
> plot(prostate)
```

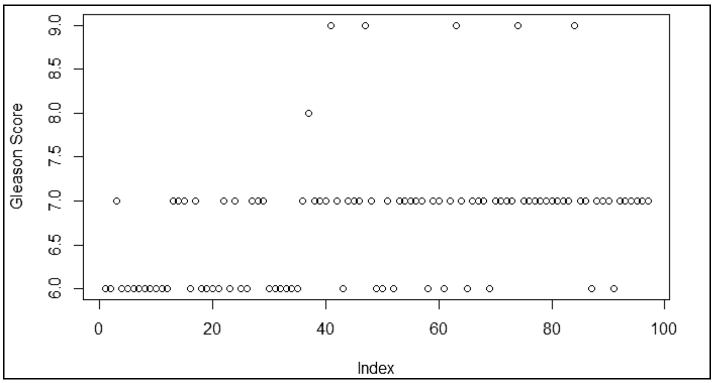
上述命令输出如下。



这么多变量放在一张图上确实有点让人搞不清状况，所以我们循序渐进。可以看出，结果变量lpsa和预测变量lcavol之间确实存在明显的线性关系。还可以看出，这些特征变量的离散度是比较合适的，而且在训练集和测试集之间的分布也比较平衡，可能的例外只有Gleason评分这个特征。请注意，这个数据集中，Gleason评分只有4个值。如果看一下位于train和gleason相交点的那个图，就会发现，有一个Gleason评分值既不属于测试集，也不输入训练集。这可能会使我们的分析过程产生问题，需要进行数据转换。所以，我们专门为这个特征建立一个统计图，如下所示：

```
> plot(prostate$gleason)
```

上述命令输出如下。



现在就看出问题了。每个点代表一个观测，X轴是数据框中观测的编号。只有1个观测的Gleason评分是8.0，只有5个观测的Gleason评分是9.0。你可以建立一个特征值表格来查看确切的数量。如下所示：

```
    > table(prostate$gleason)
 6  7  8  9
35 56  1  5
```

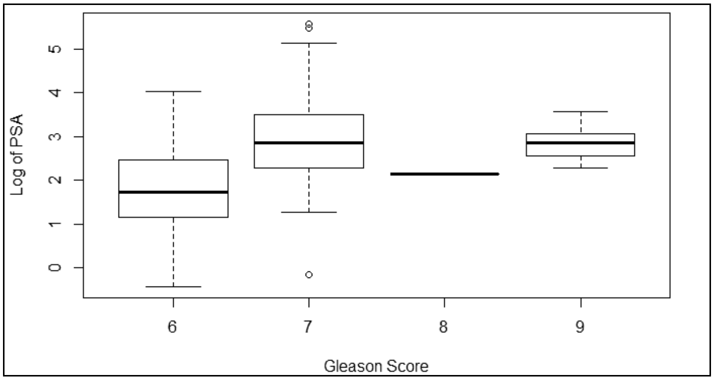
怎么办呢？我们有以下3种选择：

- ❑ 完全删除这个特征；
- ❑ 仅删除值为8.0和9.0的那些评分；
- ❑ 对特征重新编码，建立一个指标变量。

我认为，如果建立一个横轴为**Gleason Score**，纵轴为**Log of PSA**的箱线图，会对我们的选择有所帮助。在前面的章节中，我们使用ggplot2包建立箱线图，但R的基础包也可以建立箱线图，如下所示：


```
> boxplot(prostate$lpsa ~ prostate$gleason, xlab = "Gleason Score",
  ylab = "Log of PSA")
```

上述命令输出如下。



看了上面的图，我认为最好的选择是，将这个特征转换为一个指标变量，0表示评分为6，1表示评分为7或更高。删除特征可能会损失模型的预测能力。缺失值也可能会在我们将要使用的glmnet包中引起问题。

你可以非常简单地实现对指标变量的编码，通过一行代码即可。使用`ifelse()`命令，指定你想在数据框中转换的列，然后按照这个规则转换：如果观测值的特征值为*x*，则将其编码为*y*，否则将其编码为*z*。

```
> prostate$gleason <- ifelse(prostate$gleason == 6, 0, 1)
```

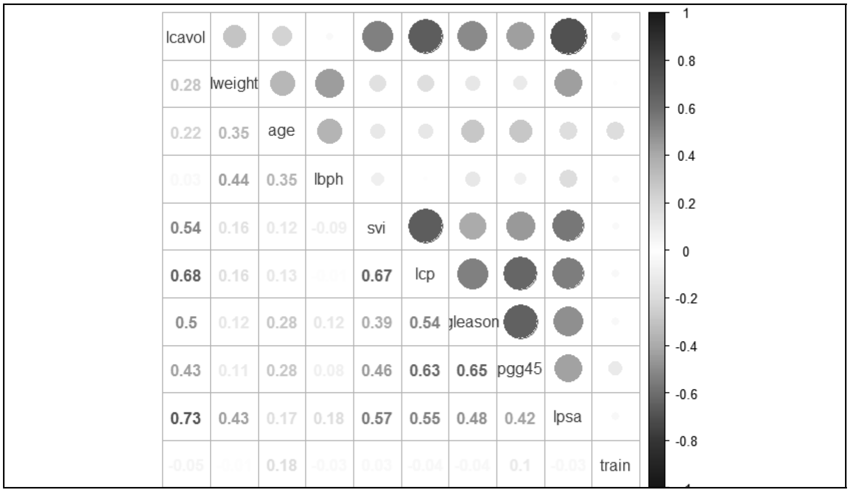
同样，我们建立一个表格，验证数据转换结果确如预想。如下所示：

```
    > table(prostate$gleason)
0   1
35  62
```

转换结果完美无缺！既然散点图矩阵比较难以理解，我们就再看看相关性统计图，它可以表示特征之间是否存在相关性或依赖。先用`cor()`函数建立一个相关性对象，然后利用corrplot库中的`corrplot.mixed()`函数做出相关性统计图，如下所示：

```
    > p.cor = cor(prostate)
> corrplot.mixed(p.cor)
```

上述命令输出如下。



这样又跳出两个问题。首先，PSA和肿瘤体积的对数（`lcavol`）高度相关。回想一下，在散点图矩阵中，它们表现出很强的线性相关关系。其次，多重共线性是个问题。例如，肿瘤体积还与包膜穿透相关，而包膜穿透还与贮精囊侵入相关。这真是一个很有趣的机器学习实战例子！

开始机器学习之前，必须先建立训练数据集和测试数据集。既然观测值中已经有一个特征指明这个观测值是否属于训练集，我们就可以使用`subset()`命令将`train`值为`TRUE`的观测值分到训练集中，将`train`值为`FALSE`的观测值分到测试集。将`train`删除也是必须的，因为我们不想把它作为预测特征。如下所示：

```
    > train <- subset(prostate, train == TRUE)[, 1:9]
> str(train)
'data.frame':67 obs.  of  9 variables:
 $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
 $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
 $ age    : int  50 58 74 58 62 50 58 65 63 63 ...
 $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ gleason: num  0 0 1 0 0 0 0 0 0 1 ...
 $ pgg45  : int  0 0 20 0 0 0 0 0 0 30 ...
 $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```
> test <- subset(prostate, train == FALSE)[, 1:9]
> str(test)
'data.frame':30 obs. of  9 variables:
 $ lcavol : num  0.737 -0.777 0.223 1.206 2.059 ...
 $ lweight: num  3.47 3.54 3.24 3.44 3.5 ...
 $ age    : int  64 47 63 57 60 69 68 67 65 54 ...
 $ lbph   : num  0.615 -1.386 -1.386 -1.386 1.475 ...
 $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.386 -1.386 -1.386 -0.431 1.348 ...
 $ gleason: num  0 0 0 1 1 0 0 1 0 0 ...
 $ pgg45  : int  0 0 0 5 20 0 0 20 0 0 ...
 $ lpsa   : num  0.765 1.047 1.047 1.399 1.658 ...
```


4.3　模型构建与模型评价

数据已经准备好了，我们将开始构建模型。为了进行对比，先用最优子集回归建立一个模型，像我们在前两章中做的那样，然后使用正则化技术建立模型。

4.3.1　最优子集

下面的代码（或者说大部分代码）基本上是我们在第2章中使用过的代码的翻版。通过regsubsets()命令建立一个最小子集对象，然后指定训练数据集。选择出的特征随后用在测试集上，通过计算均方误差来评价模型。

我们建立模型的语法为lpsa~.，使用波形符号加句号说明，要使用数据框中除响应变量之外的所有变量进行预测。如下所示：

```
> subfit <- regsubsets(lpsa ~ ., data = train)
```

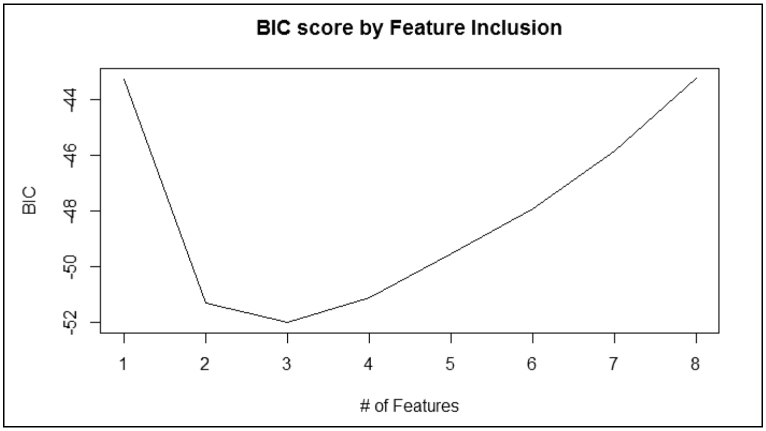
模型建立之后，你可以通过两行代码得到最优子集。第一行代码将摘要模型写入一个对象，然后从这个对象提取各个子集，使用which.min()命令确定最优子集。在本例中，我们使用第2章中讨论过的贝叶斯信息准则，如下所示：

```
    > b.sum <- summary(subfit)
> which.min(b.sum$bic)
  [1] 3
```

结果告诉我们，三特征模型具有最小的BIC值。可以通过一个统计图查看模型性能和子集组合之间的关系，如下所示：

```
> plot(b.sum$bic, type = "l", xlab = "# of Features", ylab = "BIC",
  main = "BIC score by Feature Inclusion")
```

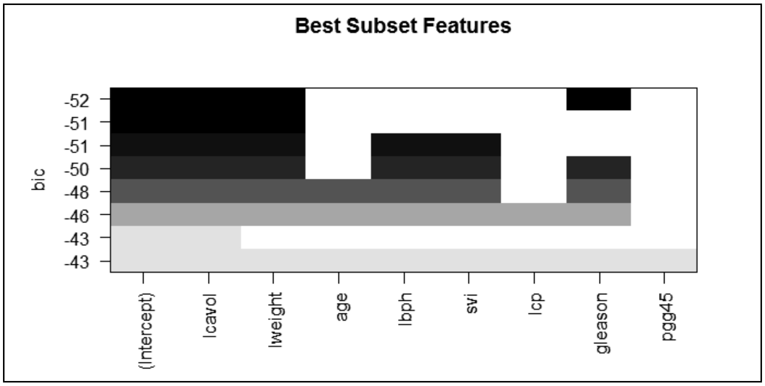
上述命令输出如下页图。



对实际模型做出统计图，可以让我们进行更详细的检查。如下所示：

```
> plot(subfit, scale = "bic", main = "Best Subset Features")
```

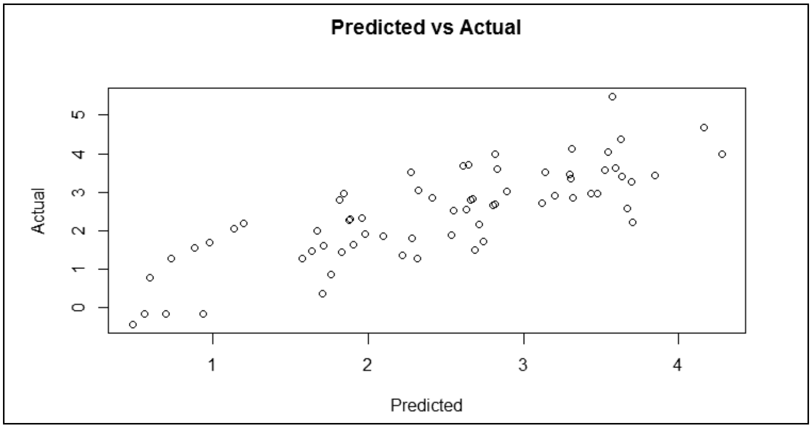
上述命令输出如下。



于是，上图告诉我们具有最小BIC值的模型中的3个特征是：`lcavol`、`lweight`和`gleason`。值得注意的是，`lcavol`包含在所有模型组合之中，这与我们之前的数据探索结果是一致的。现在，可以在测试集上试验模型了，但要先用模型的拟合值与实际值画一张图，看看最终解中的线性关系，并检查同方差性。需要用以上3个变量建立一个线性模型，因为是用OLS建立的模型，所以把它存储在名为`ols`的对象中。然后，使用OLS拟合出的模型就可以同训练集中的实际值进行对比了。如下所示：

```
    > ols <- lm(lpsa ~ lcavol + lweight + gleason, data = train)
> plot(ols$fitted.values, train$lpsa, xlab = "Predicted", ylab =
     "Actual", main = "Predicted vs Actual")
```

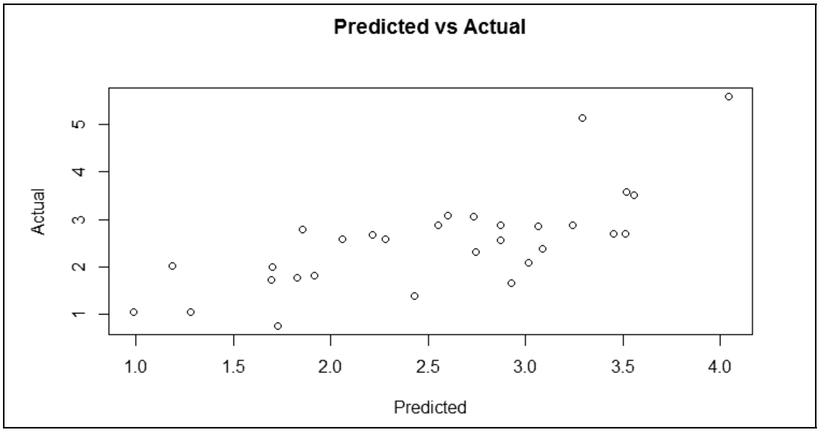
上述命令输出如下页图。



从图中可以看出，在训练集上线性拟合表现得很好，也不存在异方差性。然后看看模型在测试集上的表现，使用`predict()`函数并指定`newdata = test`，如下所示：

```
> pred.subfit <- predict(ols, newdata = test)
> plot(pred.subfit, test$lpsa , xlab = "Predicted", ylab =
    "Actual", main = "Predicted vs Actual")
```

对象中的值可以用来生成统计图，表示预测值和实际值之间的关系。如下图所示。



这个图还不是太难看。总体来说，图中呈现一种线性关系，只不过当PSA值比较高时，有两个离群点。结束本小节之前，我们需要计算**均方误差**，以便在不同模型构建技术之间进行比较。这非常简单，只要算出残差并求出残差平方的均值即可，如下所示：

```
> resid.subfit <- test$lpsa - pred.subfit
> mean(resid.subfit^2)
[1] 0.5084126
```

MSE值为0.508，以此为基准继续下面的内容。

4.3.2　岭回归

在岭回归中，我们的模型会包括全部8个特征，所以岭回归模型与最优子集模型的比较真是令人期待啊。我们要使用的程序包glmnet实际上已经加载过了。这个程序包要求输入特征存储在矩阵中，而不是在数据框中。岭回归的命令形式为`glmnet(x=输入矩阵, y=响应变量, family=分布函数, alpha=0)`。这里的alpha为0时，表示进行岭回归；alpha为1时，表示进行LASSO。

要准备好供glmnet使用的训练集数据也很容易，使用`as.matrix()`函数处理输入数据，并建立一个向量作为响应变量，如下所示：

```
    > x <- as.matrix(train[, 1:8])
> y <- train[, 9]
```

现在可以使用岭回归了，我们把结果保存在一个对象中，可以为对象起一个恰当的名字，比如ridge。这里有一点非常重要，请一定注意：glmnet包会在计算λ值之前首先对输入进行标准化，然后计算非标准化系数。你需要指定响应变量的分布为`gaussian`，因为它是连续的；还要指定`alpha = 0`，表示进行岭回归。如下所示：

```
> ridge <- glmnet(x, y, family = "gaussian", alpha = 0)
```

这个对象包含了我们进行模型评价所需的所有信息。首先尝试`print()`命令，它会展示非0系数的数量，解释偏差百分比以及相应的λ值。程序包中算法默认的计算次数是100，但如果偏差百分比在两个λ值之间的提高不是很显著的话，算法会在100次计算之前停止。也就是说，算法收敛于最优解。为了节省篇幅，我只在下面列出了前5个和后10个λ结果：

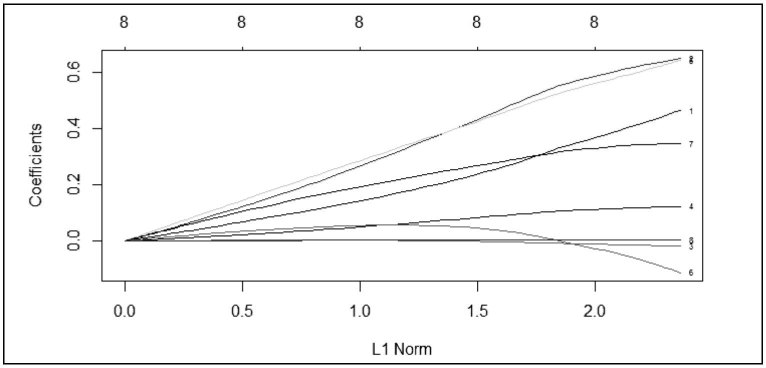
```
    > print(ridge)
Call:  glmnet(x = x, y = y, family = "gaussian", alpha = 0)
       Df       %Dev    Lambda
  [1,]  8 3.801e-36 878.90000
  [2,]  8 5.591e-03 800.80000
  [3,]  8 6.132e-03 729.70000
  [4,]  8 6.725e-03 664.80000
  [5,]  8 7.374e-03 605.80000
  ...........................
 [91,]  8 6.859e-01   0.20300
 [92,]  8 6.877e-01   0.18500
 [93,]  8 6.894e-01   0.16860
 [94,]  8 6.909e-01   0.15360
 [95,]  8 6.923e-01   0.13990
 [96,]  8 6.935e-01   0.12750
 [97,]  8 6.946e-01   0.11620
 [98,]  8 6.955e-01   0.10590
 [99,]  8 6.964e-01   0.09646
[100,]  8 6.971e-01   0.08789
```

以第100行为例。可以看出非0系数，即模型中包含的特征的数量为8。请记住，在岭回归中，这个数是不变的。还可以看出解释偏差百分比为0.6971，以及这一行的调优系数λ的值为0.08789。

此处即可决定选择在测试集上使用哪个λ。这个λ值应该是0.08789，但是为了简单起见，在测试集上可以试一下0.10。此时，一些统计图是非常有用的。我们先看看程序包中默认的统计图，设定`label = TRUE`可以给曲线加上注释，如下所示：

```
> plot(ridge, label = TRUE)
```

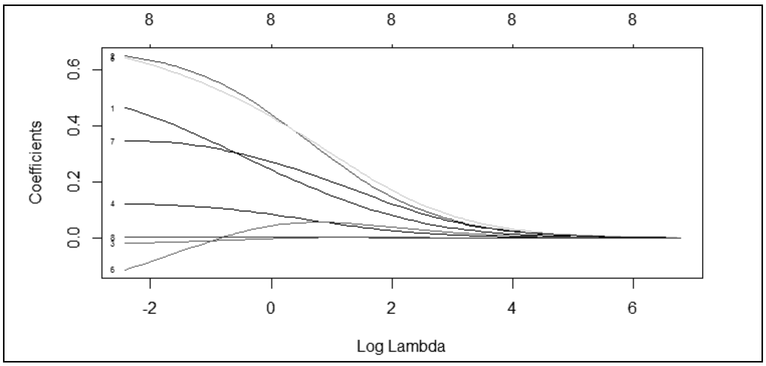
上述命令输出如下。



在默认图中，Y轴是**系数值**，X轴是**L1范数**，图中显示了系数值和L1范数之间的关系。图的上方有另一条X轴，其上的数值表示模型中的特征数。查看统计图的一种更好方式是，看系数值如何随着λ的变化而变化。只需在`plot()`命令中稍稍调整，加上参数`xvar="lambda"`即可。另一种选择是，看系数值如何随解释偏差百分比变化，将`lamda`换成`dev`即可。

```
> plot(ridge, xvar = "lambda", label = TRUE)
```

上述命令输出如下。



这张图非常有价值，因为它表明，当λ值减小时，压缩参数随之减小，而系数绝对值随之增大。要想看看当λ为一个特定值时的系数值，可以使用`coef()`命令。现在，看一下当λ为0.1时，系数值是多少。指定参数`s=0.1`，同时指定参数`exact=TRUE`，告诉glmnet在拟合模型时使用具

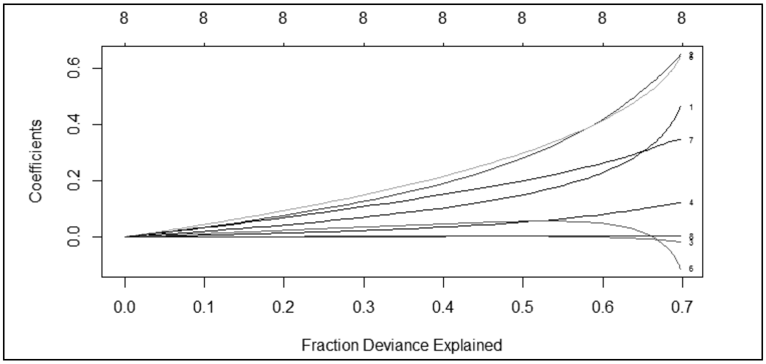
体的λ值，而不是从λ值的两侧选值插入。如下所示：

```
    > ridge.coef <- coef(ridge, s = 0.1, exact = TRUE)
> ridge.coef
9 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept)  0.13062197
lcavol       0.45721270
lweight      0.64579061
age         -0.01735672
lbph         0.12249920
svi          0.63664815
lcp         -0.10463486
gleason      0.34612690
pgg45        0.00428580
```

需要特别注意的是，age、lcp和pgg45的系数非常接近0，但还不是0。别忘了再看一下偏差与系数之间的关系图：

```
> plot(ridge, xvar = "dev", label = TRUE)
```

上述命令输出如下。



同前两张图相比，我们可以从这张图中看出，当λ减小时，系数会增大，解释偏差百分比也会增大。如果将λ值设为0，就会忽略收缩惩罚，模型将等价于OLS。

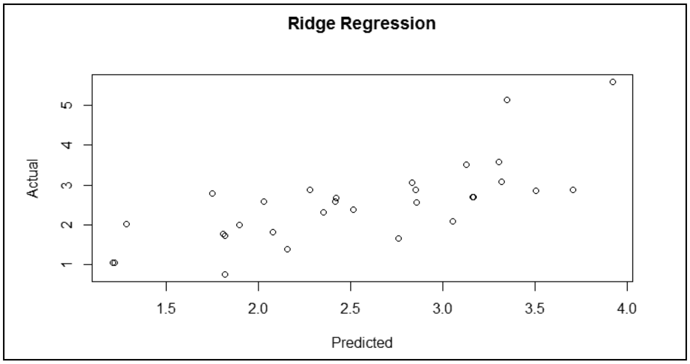
为了在测试集上证明这一点，需要转换特征，像我们在训练集上做的一样：

```
> newx <- as.matrix(test[, 1:8])
```

然后，使用predict()函数建立一个名为ridge.y的对象，指定参数type="response"以及λ值为0.10，画出表示预测值和实际值关系的统计图，如下所示：

```
    > ridge.y <- predict(ridge, newx = newx, type = "response", s =
    0.1)
> plot(ridge.y, test$lpsa, xlab = "Predicted", ylab = "Actual",main
    = "Ridge Regression")
```

上述命令输出如下。



表示岭回归中预测值和实际值关系的统计图看上去与最优子集的非常相似，同样地，在PSA测量结果比较大的一端有两个有趣的离群点。在实际情况下，我建议对离群点进行更深入的研究，搞清楚是它们真的与众不同，还是我们忽略了什么。这就是领域专家的用武之地。与MSE基准的比较可能告诉我们一些不同的事。先算出残差，然后算出残差平方的平均值：

```
    > ridge.resid <- ridge.y - test$lpsa
> mean(ridge.resid^2)
[1] 0.4789913
```

岭回归给出的MSE稍好一点E。现在是时候检验LASSO了，看看我们能否将误差再减少一些。

4.3.3 LASSO

下面运行LASSO就非常简单了，只要改变岭回归模型的一个参数即可。也就是说，在`glmnet()`语法中将`alpha=0`变为`alpha=1`。运行代码，看看模型的输出，检查前5个和后10个拟合结果：

```
    > lasso <- glmnet(x, y, family = "gaussian", alpha = 1)
> print(lasso)
Call: glmnet(x = x, y = y, family = "gaussian", alpha = 1)
Df %Dev Lambda
[1,] 0 0.00000 0.878900
[2,] 1 0.09126 0.800800
[3,] 1 0.16700 0.729700
[4,] 1 0.22990 0.664800
[5,] 1 0.28220 0.605800
........................
[60,] 8 0.70170 0.003632
[61,] 8 0.70170 0.003309
[62,] 8 0.70170 0.003015
[63,] 8 0.70170 0.002747
[64,] 8 0.70180 0.002503
[65,] 8 0.70180 0.002281
```

```
[66,] 8 0.70180 0.002078
[67,] 8 0.70180 0.001893
[68,] 8 0.70180 0.001725
[69,] 8 0.70180 0.001572
```

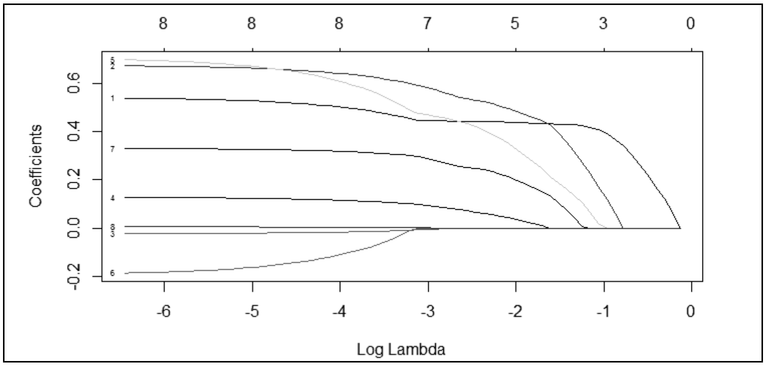
请注意，模型构建过程在69步之后停止了，因为解释偏差不再随着λ值的增加而减小。还要注意，Df列现在也随着λ变化。初看上去，当λ值为0.001572时，所有8个特征都应该包括在模型中。然而，出于测试的目的，我们先用更少特征的模型进行测试，比如7特征模型。从下面的结果行中可以看到，λ值大约为0.045时，模型从7个特征变为8个特征。因此，使用测试集评价模型时要使用这个λ值。如下所示：

```
[31,] 7 0.67240 0.053930
[32,] 7 0.67460 0.049140
[33,] 7 0.67650 0.044770
[34,] 8 0.67970 0.040790
[35,] 8 0.68340 0.037170
```

和岭回归一样，可以在图中画出结果。如下所示：

```
> plot(lasso, xvar = "lambda", label = TRUE)
```

上述命令输出如下。



这张图很有趣，真正展示了LASSO是如何工作的。请注意标号为8、3和6的曲线的表现，这几条曲线分别对应特征`pgg45`、`age`和`lcp`。看上去`lcp`一直接近于0，直到作为最后一个特征被加入模型。可以通过与岭回归中同样的操作看看7特征模型的系数值，将λ值放入`coef()`函数，如下所示：

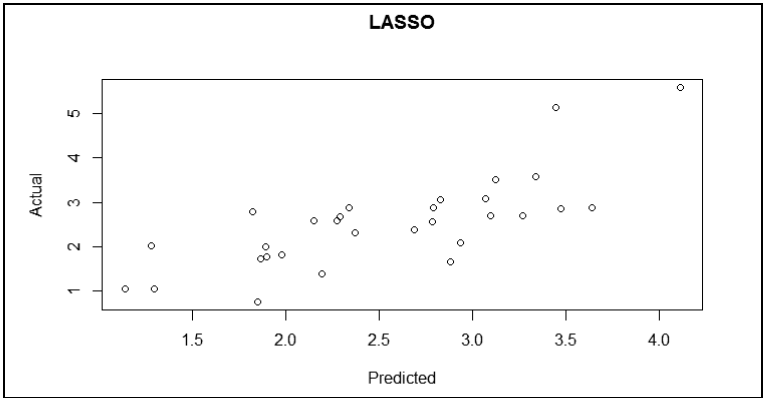
```
    > lasso.coef <- coef(lasso, s = 0.045, exact = TRUE)
> lasso.coef
9 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) -0.1305852115
lcavol       0.4479676523
lweight      0.5910362316
age         -0.0073156274
```

```
lbph        0.0974129976
svi         0.4746795823
lcp         .
gleason     0.2968395802
pgg45       0.0009790322
```

LASSO算法在λ值为0.045时，将`lcp`的系数归零。下面是LASSO模型在测试集上的表现：

```
> lasso.y <- predict(lasso, newx = newx, type = "response", s =
  0.045)
> plot(lasso.y, test$lpsa, xlab = "Predicted", ylab = "Actual",
    main = "LASSO")
```

上述命令输出如下。



像以前一样，算出MSE的值：

```
> lasso.resid <- lasso.y - test$lpsa
> mean(lasso.resid^2)
[1] 0.4437209
```

看起来我们的统计图和前面一样，只是MSE值有了一点点改进，重大改进的最后希望只能寄托在弹性网络上面了。要进行弹性网络建模，还可以继续使用glmnet包，要做的调整是不但要解出λ值，还要解出弹性网络参数α。回忆一下，$\alpha = 0$表示岭回归惩罚，$\alpha = 1$表示LASSO惩罚，弹性网络参数为$0 \leqslant \alpha \leqslant 1$。同时解出两个不同的参数会非常麻烦，令人心生怯意，但是，我们可以求助于R中的老朋友——caret包。

4.3.4 弹性网络

caret包旨在解决分类问题和训练回归模型，它配有一个很棒的网站，帮助人们掌握其所有功能：http://topepo.github.io/caret/index.html。这个软件包有很多功能可以使用，其中一些会在后面的章节中用到。现在的目的集中于找到λ和弹性网络混合参数α的最优组合。可以通过下面3个简单的步骤完成。

(1) 使用R基础包中的`expand.grid()`函数，建立一个向量存储我们要研究的α和λ的所有组合。

(2) 使用caret包中的`trainControl()`函数确定重取样方法，像第2章一样，使用LOOCV。

(3) P在caret包的`train()`函数中使用`glmnet()`训练模型来选择α和λ。

一旦选定参数，我们会像在岭回归和LASSO中做的那样，在测试数据上使用它们。

TIP 我们的组合网格应该足够大，以便能获得最优模型；但又不能太大，导致计算上不可行。对于本章这种规模的数据集，不用担心会出现这样的问题，但一定要时刻牢记。

可以按照下面的规则试验这两个超参数。

- α从0到1，每次增加0.2；请记住，α被绑定在0和1之间。
- λ从0到0.20，每次增加0.02；0.2的λ值是岭回归λ值（$\lambda=0.1$）和LASSOλ值（$\lambda=0.045$）之间的一个中间值。

可以使用`expand.grid()`函数建立这个向量并生成一系列数值，caret包会自动使用这些数值。caret包通过下列代码生成α值和λ值：

```
> grid <- expand.grid(.alpha = seq(0, 1, by = .2), .lambda =
  seq(0.00, 0.2,  by = 0.02))
```

使用`table()`函数，可以看到α和λ的全部66种组合：

```
    > table(grid)
      .lambda
.alpha 0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2
   0   1    1    1    1    1   1    1    1    1    1   1
   0.2 1    1    1    1    1   1    1    1    1    1   1
   0.4 1    1    1    1    1   1    1    1    1    1   1
   0.6 1    1    1    1    1   1    1    1    1    1   1
   0.8 1    1    1    1    1   1    1    1    1    1   1
   1   1    1    1    1    1   1    1    1    1    1   1
```

可以确认这就是我们想要的结果——α值在0和1之间，λ值在0和0.2之间。

对于重取样方法，我们要在代码中将method参数指定为`LOOCV`。还有其他的重取样方法可以选择，比如自助法和K折交叉验证法。`trainControl()`函数中有更多选择，我们会在后续章节进行研究。

在`trainControl()`函数中，你还可以通过`selectionFunctioin()`函数指定模型选择方法。对于定量型响应变量，使用算法的默认选择**均方根误差**即可完美实现：

```
> control <- trainControl(method = "LOOCV")
```

现在可以使用train()函数确定最优的弹性网络参数了。这个函数和lm()很相似，只需在函数语法中加上method="glmnet", trControl=control, tuneGrid=grid。将结果存储在一个名为enet.train的对象中：

```
> enet.train <- train(lpsa ~ ., data = train, method = "glmnet",
  trControl = control, tuneGrid = grid)
```

调用这个对象，可以看到能够得出最小RMSE值的参数组合，如下所示：

```
    > enet.train
glmnet
67 samples
 8 predictor
No pre-processing
Resampling:
Summary of sample sizes: 66, 66, 66, 66, 66, 66, ...
Resampling results across tuning parameters:
  alpha  lambda  RMSE  Rsquared
  0.0    0.00    0.750 0.609
  0.0    0.02    0.750 0.609
  0.0    0.04    0.750 0.609
  0.0    0.06    0.750 0.609
  0.0    0.08    0.750 0.609
  0.0    0.10    0.751 0.608
   ........................
  1.0    0.14    0.800 0.564
  1.0    0.16    0.809 0.558
  1.0    0.18    0.819 0.552
  1.0    0.20    0.826 0.549
```


我们选择最优模型的原则是RMSE值最小，模型最后选定的最优参数组合是$\alpha = 0$，$\lambda = 0.08$。

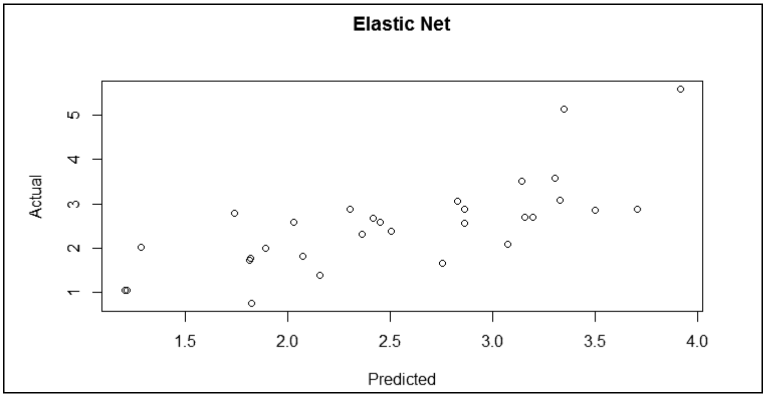
实验设计得到的最优调优参数是$\alpha = 0$和$\lambda = 0.08$，相当于glmnet中$s = 0.08$的岭回归。回忆一下，我们在4.3.2节中得出的λ是0.10。R方为61%，真是乏善可陈。

在测试集上验证模型的过程和前面一样：

```
    > enet <- glmnet(x, y, family = "gaussian", alpha = 0, lambda =
      .08)
> enet.coef <- coef(enet, s = .08, exact = TRUE)
> enet.coef
9 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept)  0.137811097
lcavol       0.470960525
lweight      0.652088157
age         -0.018257308
lbph         0.123608113
svi          0.648209192
lcp         -0.118214386
gleason      0.345480799
```

```
pgg45         0.004478267
> enet.y <- predict(enet, newx=newx, type="response", s=.08)
> plot(enet.y, test$lpsa, xlab="Predicted", ylab="Actual",
      main="Elastic Net")
```

上述命令输出如下。



和以前一样，计算MSE：

```
    > enet.resid <- enet.y - test$lpsa
> mean(enet.resid^2)
[1] 0.4795019
```

这个模型的误差与岭回归很接近。在测试集上，LASSO模型在误差方面表现最好。模型可能过拟合了！我们的三特征最优子集模型是最容易解释的，但考虑误差的话，却更应该接收另外一种技术得出的模型。可以在glmnet包中使用10折交叉验证来确定哪个模型更好。

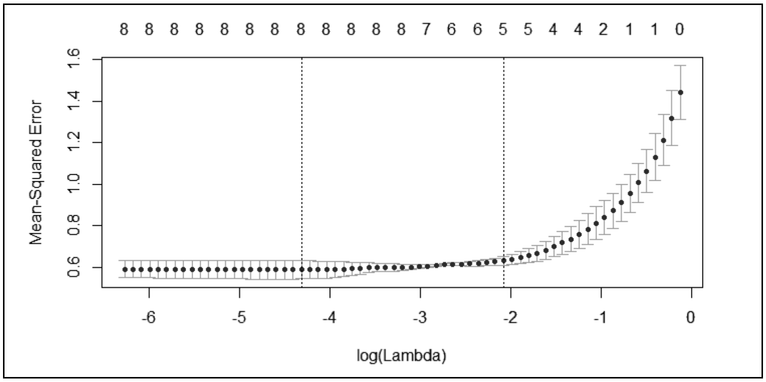
4.3.5　使用 glmnet 进行交叉验证

我们通过caret包使用过LOOCV，现在试试K折交叉验证。glmnet包在使用`cv.glmnet()`估计λ值时，默认使用10折交叉验证。在K折交叉验证中，数据被划分成k个相同的子集（折），每次使用$k-1$个子集拟合模型，然后使用剩下的那个子集做测试集，最后将k次拟合的结果综合起来（一般取平均数），确定最后的参数。

在这个方法中，每个子集只有一次用作测试集。在glmnet包中使用K折交叉验证非常容易，结果包括每次拟合的λ值和响应的MSE。默认设置为$\alpha=1$，所以如果你想试试岭回归或弹性网络，必须指定α值。因为我们想看看尽可能少的输入特征的情况，所以还是使用默认设置，但由于训练集中数据量的原因，只分3折：

```
    > set.seed(317)
> lasso.cv = cv.glmnet(x, y, nfolds = 3)
> plot(lasso.cv)
```

上述代码的输出如下。



CV统计图和glmnet中其他统计图有很大区别，它表示λ的**对数值和均方误差**之间的关系，还带有模型中特征的数量。图中两条垂直的虚线表示取得MSE最小值的logλ（左侧虚线）和距离最小值一个标准误差的logλ。如果有过拟合问题，那么距离最小值一个标准误的位置是非常好的解决问题的起点。你还可以得到这两个λ的具体值，如下所示：

```
> lasso.cv$lambda.min #minimum
[1] 0.0133582
> lasso.cv$lambda.1se #one standard error away
[1] 0.124579
```

使用lambda.1se可以完成下面的过程，查看系数并在测试集上进行模型验证：

```
> coef(lasso.cv, s = "lambda.1se")
9 x 1 sparse Matrix of class "dgCMatrix"
 1
(Intercept) -0.13543760
lcavol 0.43892533
lweight 0.49550944
age .
lbph 0.04343678
svi 0.34985691
lcp .
gleason 0.21225934
pgg45 .

> lasso.y.cv = predict(lasso.cv, newx=newx, type = "response",
    s = "lambda.1se")

> lasso.cv.resid = lasso.y.cv - test$lpsa

> mean(lasso.cv.resid^2)
[1] 0.4465453
```

这个模型的误差为0.45，只有5个特征，排除了`age`、`lcp`和`pgg45`。

4.4　模型选择

通过对数据集的分析和研究，我们得出5个不同模型。下面是这些模型在测试集上的误差。

- 最优子集模型：0.51
- 岭回归模型：0.48
- LASSO模型：0.44
- 弹性网络模型：0.48
- LASSO交叉验证模型：0.45

仅看误差的话，7特征LASSO模型表现最好。但是，这个最优模型能解决我们试图回答的问题吗？我们通过交叉验证得到λ值约为0.125的模型，它更简约，也可能更加合适。我更倾向于选择它，因为其解释性更好。

说到这里，显然需要来自肿瘤专家、泌尿科专家和病理学家的专业知识，来帮助我们搞清什么是最有意义的。确实如此，但同时也需要更多数据。在本例的样本规模之下，仅改变随机数种子或重新划分训练集和测试集都可能使结果发生大的改变（你可以试试）。到头来，这些结果非但不能提供答案，还可能引起更多问题。但是，这很糟糕吗？当然不是，除非在项目开始之初你就犯下了严重的错误，对你能做的事情夸下海口。我们在第1章就提出了合理的警告，要审慎稳妥地推进你的任务。

4.5　正则化与分类问题

上面使用的正则化技术同样适用于分类问题，二值分类和多值分类皆可。因此，结束本章之前，我们再介绍一下可以用于逻辑斯蒂回归问题的示例代码。更具体地说，是可以用于上一章乳腺癌数据集的代码。在具有定量型响应变量的回归问题中，正则化是一种处理高维数据集的重要技术。

逻辑斯蒂回归示例

回忆一下，在我们分析过的乳腺癌数据中，肿瘤是恶性的概率可以用逻辑斯蒂函数表示如下：

$$\mathrm{P}(\mathrm{malignant}) = 1/1 + e^{-(B_0\ +\ B_1X_1\ +\ B_nX_n)}$$

因为这个函数中有线性的部分，所以可以使用L1和L2正则化。和前一章一样，先加载并准备好乳腺癌数据：

```
> library(MASS)
> biopsy$ID = NULL
> names(biopsy) = c("thick", "u.size", "u.shape", "adhsn",
```

```
"s.size", "nucl", "chrom", "n.nuc", "mit", "class")
> biopsy.v2 <- na.omit(biopsy)
> set.seed(123)
> ind <- sample(2, nrow(biopsy.v2), replace = TRUE, prob = c(0.7,
  0.3))
> train <- biopsy.v2[ind==1, ]
> test <- biopsy.v2[ind==2, ]
```

转换数据，生成输入矩阵和标签：

```
> x <- as.matrix(train[, 1:9])
> y <- train[, 10]
```

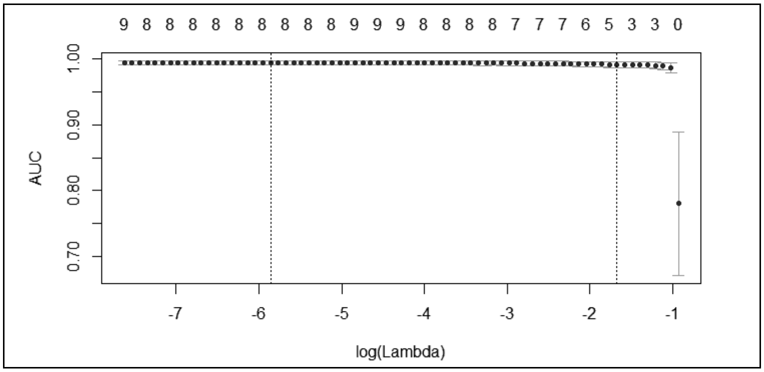

在函数`cv.glmnet`中，将`family`的值设定为`binomial`，将`measure`的值设定为曲线下面积（auc），并使用5折交叉验证：

```
> set.seed(3)
> fitCV <- cv.glmnet(x, y, family = "binomial",
     type.measure = "auc",
     nfolds = 5)
```

绘制fitCV，可以看出AUC和λ的关系：

```
> plot(fitCV)
```

绘图命令输出如下。



非常有意思！仅加入一个特征就可以使AUC有立竿见影的提高。下面看看在一个标准误差之处的模型系数：

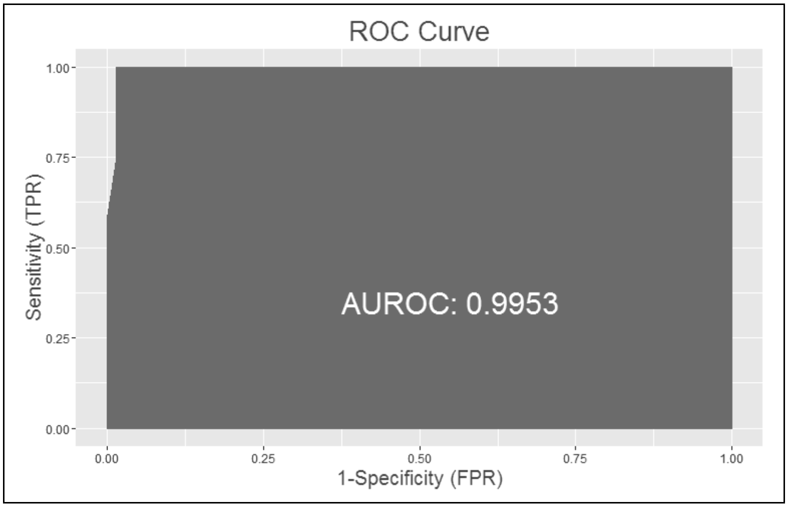
```
> fitCV$lambda.1se
[1] 0.1876892
> coef(fitCV, s = "lambda.1se")
10 x 1 sparse Matrix of class "dgCMatrix"
 1
(Intercept) -1.84478214
thick 0.01892397
u.size 0.10102690
```

```
u.shape 0.08264828
adhsn .
s.size .
nucl 0.13891750
chrom .
n.nuc .
mit .
```

可以看出，选择出的4个特征是thickness、u.size、u.shape和nucl。和前一章一样，通过误差和auc，我们看看这个模型在测试集上的表现：

```
> library(InformationValue)
> predCV <- predict(fitCV, newx = as.matrix(test[, 1:9]),
    s = "lambda.1se",
    type = "response")
actuals <- ifelse(test$class == "malignant", 1, 0)
misClassError(actuals, predCV)
[1] 0.0622
> plotROC(actuals, predCV)
```

上述代码输出如下。



结果显示，这个模型的效果与前面的逻辑斯蒂回归模型基本一样。看上去，lambda.1se还不是最优的选择。我们看看使用lambda.min选择的模型是否可以再次改善样本预测结果：

```
> predCV.min <- predict(fitCV, newx = as.matrix(test[, 1:9]),
    s = "lambda.min",
    type = "response")
> misClassError(actuals, predCV.min)
[1] 0.0239
```

就是它了！这个错误率和第3章中的一样低。

4.6　小结

本章的目标是，通过一个小数据集介绍如何对线性模型应用高级特征选择技术。数据集的结果变量是定量的，但我们使用的glmnet包也支持定性的结果变量（二值分类和多值分类）。我们介绍了正则化及其包含的3种技术，并应用这些技术构建模型，然后进行了比较。正则化是一项强大的技术，与其他建模技术相比，既可以提高计算效率，还可以提取更有意义的特征。此外，我们还开始使用caret包在训练模型时使多个参数达到最优化。直到现在，我们还是仅讨论了线性模型。接下来的两章中，我们会开始使用非线性模型解决分类问题和回归问题。

第 5 章

更多分类技术：K最近邻与支持向量机

“统计思维总有一天会像读与写一样，成为一个有效率公民的必备能力。”

——赫伯特·乔治·威尔斯

我们在第3章讨论了逻辑斯蒂回归，它被用来预测一个观测属于某个响应变量分类的概率——我们称之为分类问题。逻辑斯蒂回归只是分类方法的开始，还可以使用很多其他方法改善预测质量。

在本章中，我们将深入研究两种非线性技术：**K最近邻**（KNN）与**支持向量机**（SVM）。这两种技术要比我们之前讨论的那些技术复杂一些，因为放弃了线性假设。也就是说，不再必须使用特征的线性组合来定义决策边界。先提醒各位，这样不一定能得到更好的预测结果，而且向业务伙伴解释模型也会有一点问题，计算效率也更低。正确使用这些技术时，可以作为本书中其他技术和工具的强有力的补充。除了能够解决分类问题之外，它们还可以用于预测连续型的结果。在本章，我们将集中讨论分类问题。

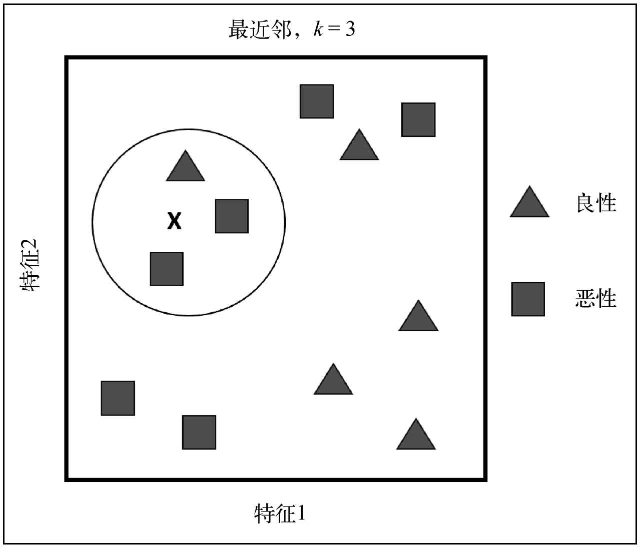
介绍这两种技术的高深背景知识之后，我们提供一个商业案例，然后应用这两种技术得到最优模型。先从KNN开始。

5.1　K 最近邻

在之前的工作中，我们建立的模型具有系数。换句话说，要对模型包含的特征进行参数估计。KNN中没有参数，因为这种学习方法是所谓的“基于实例的学习”。简言之，保存被标记过的实例（输入和相应的输出标记），什么都不做，直至一个新的输入模式请求一个输出值。（Battiti和Brunato，2014，p.11）。这种方法通常称为**懒惰学习**，因为不产生具体的模型参数。用于训练的实例本身就是知识。要预测任何一个新实例（新数据点）时，需要对训练数据进行搜索，找到一个最类似于新实例的实例。KNN处理分类问题的方法是找最近的点（最近邻）来确定正确的分类。

k的作用是确定算法应该检查多少个近邻，如果$k = 5$，算法将检查5个最近的点。这种方法的缺点是，所有5个点在算法中都被赋予相同的权重——尽管它们在学习过程中的相关性不高。我们使用R实现这种方法，并尽力克服这种不足。

给出一个二值分类学习问题的可视化例子，这是理解K最近邻方法的最好方式。下图有两个特征可以用来预测肿瘤是“良性”还是“恶性”。图中的X表示我们要预测的新观测。如果算法设定$k = 3$，那么圆中包含的3个观测就是我们想进行评分的观测的最近邻。因为其中占多数比例的分类是“恶性”，所以X数据点被分类为“恶性”。如下图所示。



这个例子虽然简单，但也可以明显看出k的选择对于最近邻至关重要。如果k太小，那么测试集上的观测可能会有很高的方差——尽管偏差很低。另一方面，当k增加时，方差会减小，但偏差可能会变得不可接受。必须进行交叉验证以确定合适的k值。

另一个需要指出的重要问题是距离的计算，或者说是特征空间中数据点的临近度的计算。默认的距离是**欧氏距离**，也就是从点A到点B的简单直线距离——就像乌鸦飞过的直线。你也可以使用公式计算，欧氏距离等于两点坐标之差的平方和的平方根。给定两点A和B，坐标分别为p_1、p_2、…、p_n和q_1、q_2、…、q_n，欧氏距离的公式如下所示：

$$\text{欧氏距离}(\mathrm{A}, \mathrm{B}) = \sqrt{\sum\nolimits_{i=1}^{n} (p_i - q_i)^2}$$

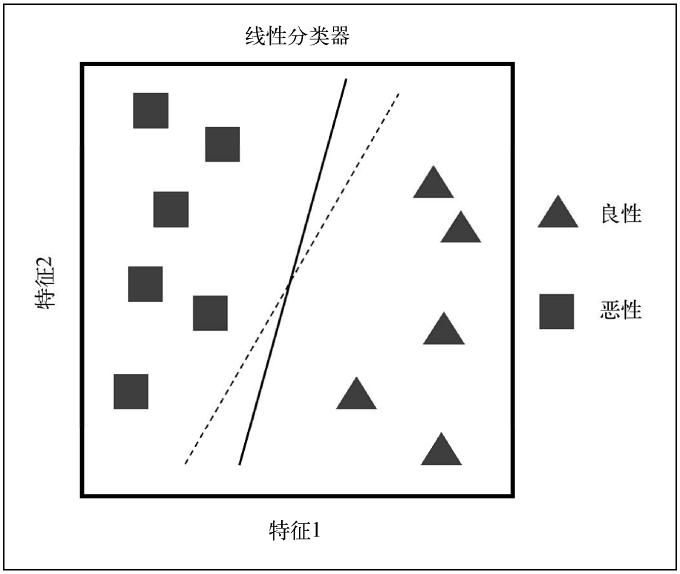
两点间的距离强烈依赖于测量特征时使用的单位，所以必须对其进行标准化。除了这种与距离相关的权重以外，还可以使用距离计算的其他方式。在接下来的例子中，我们会研究这个问题。

5.2　支持向量机

必须承认，第一次听说支持向量机的时候，我真被弄糊涂了，认为这是某种形式的学术恶搞或是某个圈子的内部笑话。但是，经过对SVM的虚心学习，我对这项技术的态度已经从半信半疑变为推崇备至。

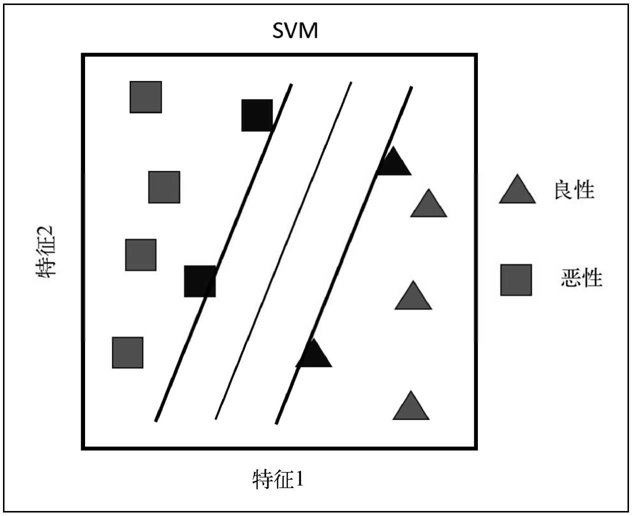
人们已经在很多情况下证明SVM表现优异，它经常被认为是最好的“开箱即用”的分类器之一。（James，G.，2013）。为了切实理解这个主题，我们再看一个简单的可视化例子。下图中的分类任务是线性可分的，但虚线和实线只不过是无数线性可行解中的两个。

如果问题的维度多于2，你就需要分割超平面。



太多的解对于泛化会是个问题，因为不论你选择哪个解，线右边的观测值都会被分到“良性”类中，线左边的观测值都会被分到“恶性”类。所以，所有线在训练数据上都是无偏的，但是对于测试数据却可能出现大相径庭的误差。这就是必须要有支持向量的原因。我们考察一下任意一个点落入线性分类器错误一侧的概率，对于虚线来说，这个概率要高于实线，这说明实线具有更高的分类安全边际。所以，正如Battiti和Brunato所说，SVM是具有最大可能边际的线性分类器，支持向量就是最接近安全边际两侧的那些向量。

下页图说明了这个思想。细实线就是最优线性分类器，它建立了上面提到的最大可能边际，提高了一个新观测落入分类器正确一侧的概率。两条粗黑线对应着安全边际，阴影数据点构成了支持向量。如果支持向量发生移动，就会导致边际和决策范围发生改变。分类器之间的距离被称为**边际**。



这一切简直完美，但是现实世界中的问题并不是这么泾渭分明的。

如果数据不是线性可分的，很多观测值就会落到分类边际错误的一侧（所谓**松弛变量**），这就是误分类。建立SVM算法的关键是，通过交叉验证找出最优数量的支持向量。任何一个正好位于最大分类边际上的观测都可以被认为是**支持向量**。

如果误差值的调优参数过大，你就会找到很多支持向量，受到高偏差低方差的困扰。而如果调优参数过小，就会出现相反的情况。按照James等人（他用C代表调优参数）的说法，当C减小时，对观测处于边际错误一侧的容忍度降低，边际变窄。这个C（更确切地说是“成本函数”）允许观测位于边际错误的一侧。如果C被设为0，则不允许任何观测违反边际。

SVM中的另一个重要问题是处理非线性模型的能力，非线性模型的输入特征带有二次项或更高阶的多项式。在SVM中，这种处理被称为**核技巧**。高阶特征可以通过交叉验证进行估计和选择，此处要介绍一种新的方法。

对于任何模型，你都可以通过不同阶数的多项式、交互项或其他衍生项来扩展特征的数量。在大规模数据集中，这样做可能失控。SVM中的核技巧可以使我们有效扩展特征空间，目的是使特征空间近似于线性可分。

怎么做到这一点呢？先来看看SVM最优化问题及其约束条件。我们希望达到以下目标。

- 找出使边际最大的权值。
- 满足约束条件：没有（或尽量少的）数据点位于边际之内。

在线性回归中，权重要与每个观测相乘。但在SVM中，权重只作用于作为支持向量的观测的内积。

这是什么意思呢？两个向量的内积就是它们对应分量的乘积之和。例如，如果向量一为[3, 4, 2]，向量二为[1, 2, 3]，那么内积就是$3\times1+4\times2+2\times3$，也就是17。在SVM中，如果要算出任意一个观测和所有其他观测之间的内积，那么算式的数量就会有$n(n-1)/2$个。n是观测的数量，有10个观测就能算出45个内积。但是，SVM只关心作为支持向量的观测及其权重。对于一个线性SVM分类器，其公式如下：

$$f(x)=\beta_0+\sum_{i=1}^{n}\alpha(x,x_i)$$

这里的(x, x_i)是支持向量的内积，因为只有当观测是支持向量时，α才不为0。

这就使分类算法中的项大大减少，从而可以应用核函数——通常就是指核技巧。

核函数的巧妙之处在于，它将特征到高维空间的转换进行了数学上的简化，不需要在高维空间中显式地创建特征。这样做的好处是，在建立高维非线性空间和决策边界的同时，还能保持最优问题的计算有效性。核函数不用将特征转换到高维空间即可计算特征在高维空间中的内积。

一般用特征的内积（点积）表示核函数，用x_i和x_j代表向量，γ和c为参数，常用的核函数如下所示。

- 线性核函数：$K(x_i, x_j)=x_i\cdot x_j$，无需转换。
- 多项式核函数：$K(x_i, x_j)=(\gamma x_i\cdot x_j+c)^d$，$d$为多项式的次数。
- 径向基核函数：$K(x_i, x_j)=e(-\gamma|x_i-x_j|^2)$。
- sigmod核函数：$K(x_i, x_j)=\tanh(\gamma x_i\cdot x_j+c)$。

至于如何在非线性技术中选择核函数，可能需要反复实验，我们会简单介绍各种选择技术。

5.3 商业案例

在接下来的案例研究中，我们在同一个数据集上应用KNN和SVM，这可以比较解决同一问题的R代码和学习方法。我们从KNN开始，还会花一些时间对混淆矩阵进行深入研究，并对评价模型正确率的各个统计量进行比较。

5.3.1 业务理解

我们要研究的数据来自美国国家糖尿病消化病肾病研究所，这个数据集包括532个观测，8个输入特征以及1个二值结果变量（Yes/No）。这项研究中的患者来自美国亚利桑那州中南部，是皮玛族印第安人的后裔。数据显示，在过去的30年中，科学家已经通过研究证明肥胖是引发糖尿病的重要因素。选择皮玛印第安人进行这项研究是因为，半数成年皮玛印第安人患有糖尿病。而

这些患有糖尿病的人中，有95%超重。研究仅限于成年女性，病情则按照世界卫生组织的标准进行诊断，为Ⅱ型糖尿病。这种糖尿病的患者胰腺功能并未完全丧失，还可以产生胰岛素，因此又称“非胰岛素依赖型”糖尿病。

我们的任务是研究那些糖尿病患者，并对这个人群中可能导致糖尿病的风险因素进行预测。久坐不动的生活方式和高热量的饮食习惯使得糖尿病已经成为美国的流行病。根据美国糖尿病协会的数据，2010年，糖尿病成为美国排名第七的致死疾病，这个结果还不包括那些未被诊断出来的病例。糖尿病还会大大增加其他疾病的发病概率，比如高血压、血脂异常、中风、眼疾和肾脏疾病。糖尿病及其并发症的医疗成本非常巨大，据估计，美国2012年糖尿病治疗总成本大约为4900亿美元。要想知道这个问题的更多背景知识，请参考美国糖尿病协会网站http://diabetes.org/diabetes-basics/statistics/。

5.3.2 数据理解和数据准备

数据集包含了532位女性患者的信息，存储在两个数据框中。数据集变量如下。

- `npreg`：怀孕次数
- `glu`：血糖浓度，由口服葡萄糖耐量测试给出
- `bp`：舒张压（单位为mm Hg）
- `skin`：三头肌皮褶厚度（单位为mm）
- `bmi`：身体质量指数
- `ped`：糖尿病家族影响因素
- `age`：年龄
- `type`：是否患有糖尿病（是/否）

数据集包含在MASS这个R包中，一个数据框是Pima.tr，另一个数据框的是Pima.te。我们不将它们分别作为训练集和测试集，而是将其合在一起，然后建立自己的训练集和测试集，目的是学习如何使用R完成这样的任务。

首先加载下面的程序包，练习时会用到它们：

```
    > library(class) #k-nearest neighbors
> library(kknn) #weighted k-nearest neighbors
> library(e1071) #SVM
> library(caret) #select tuning parameters
> library(MASS) # contains the data
> library(reshape2) #assist in creating boxplots
> library(ggplot2) #create boxplots
> library(kernlab) #assist with SVM feature selection
```

现在加载数据集并检查其结构，确保结构相同。从Pima.tr开始，如下所示：

```
> data(Pima.tr)
> str(Pima.tr)
'data.frame':200 obs. of  8 variables:
 $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
 $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
 $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
 $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
 $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3
     ...
 $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
 $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
 $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
> data(Pima.te)
> str(Pima.te)
'data.frame':332 obs. of  8 variables:
 $ npreg: int  6 1 1 3 2 5 0 1 3 9 ...
 $ glu  : int  148 85 89 78 197 166 118 103 126 119 ...
 $ bp   : int  72 66 66 50 70 72 84 30 88 80 ...
 $ skin : int  35 29 23 32 45 19 47 38 41 35 ...
 $ bmi  : num  33.6 26.6 28.1 31 30.5 25.8 45.8 43.3 39.3 29 ...
 $ ped  : num  0.627 0.351 0.167 0.248 0.158 0.587 0.551 0.183
      0.704 0.263 ...
 $ age  : int  50 31 21 26 53 51 31 33 27 29 ...
 $ type : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 2 1 1 2 ...
```

检查数据集结构之后，我们确信可以把两个数据框合成一个。这非常容易，使用`rbind()`函数即可，它的功能是绑定行并追加数据。如果你的每个数据框都有相同的观测并想追加特征，则应该使用`cbind()`函数按列绑定数据。给你的新数据框命名也非常简单，可以使用语法`new data=rbind(data frame1, data fram2)`。代码如下：

```
> pima <- rbind(Pima.tr, Pima.te)
```

一如既往，再次检查数据集结构，确认没有问题。如下所示：

```
> str(pima)
'data.frame':532 obs. of 8 variables:
 $ npreg: int 5 7 5 0 0 5 3 1 3 2 ...
 $ glu : int 86 195 77 165 107 97 83 193 142 128 ...
 $ bp : int 68 70 82 76 60 76 58 50 80 78 ...
 $ skin : int 28 33 41 43 25 27 31 16 15 37 ...
 $ bmi : num 30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3
   ...
 $ ped : num 0.364 0.163 0.156 0.259 0.133 ...
 $ age : int 24 55 35 26 23 52 25 24 63 31 ...
 $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

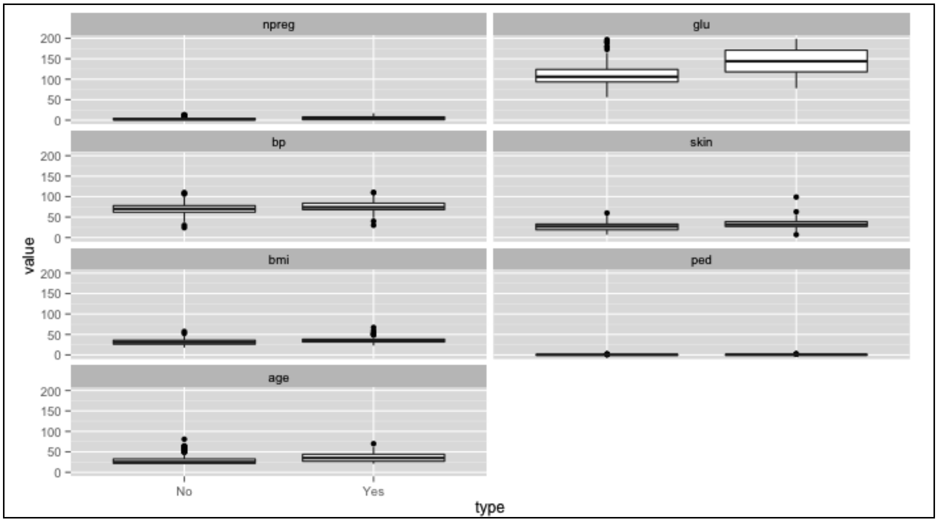
通过箱线图进行探索性分析。为此，要使用结果变量`"type"`作为ID变量的值。和逻辑斯蒂回归一样，`melt()`函数会融合数据并准备好用于生成箱线图的数据框。我们将新的数据框命名为pima.melt，如下所示：

```
> pima.melt <- melt(pima, id.var = "type")
```

使用ggplot包对箱线图进行布局是非常有效的。在ggplot()函数中,我们要指定使用的数据、x变量和y变量、统计图类型，还要说明生成的是两列统计图。在下面的代码中，将响应变量作为aes()函数中的x，响应变量的值作为y，然后使用geom_boxplot()函数生成统计图。最后，使用facet_wrap()函数将统计图分两列显示：

```
> ggplot(data = pima.melt, aes(x = type, y = value)) +
    geom_boxplot() + facet_wrap(~ variable, ncol = 2)
```

上述命令输出如下。



这张图很有趣，因为很难从中发现任何明显区别，除了血糖浓度（glu）。正如你的猜想，如果患者的血糖浓度快速升高，那么一般可以确诊为糖尿病。这里最大的问题是，不同统计图的单位不同，但却共用一个Y轴。对数据进行标准化处理并重新做图，可以解决这个问题，并生成更有意义的统计图。R有一个内建函数scale()，可以将数据转换为均值为0、标准差为1的标准形式。我们把经过标准化处理后的数据放到pima.scale新数据框，要对所有特征进行转换，只留下响应变量type。强调一下，进行KNN时，使所有特征具有同样的测量标准是很重要的。也就是说，要对数据进行标准化处理，使其均值为0，标准差为1。如果不进行标准化，那么对最近邻的距离计算就会出现错误。如果一个特征的测量标准是1~100，那么和另一个测量标准为1~10的特征相比，肯定会对结果有更大的影响。请记住，如果你对一个数据框应用了scale()函数，它就自动变成一个矩阵。使用as.data.frame()函数，将其重新变回数据框，如下所示：

```
    > pima.scale <- data.frame(scale(pima[, -8]))
> str(pima.scale)
'data.frame':532 obs. of 7 variables:
 $ npreg: num 0.448 1.052 0.448 -1.062 -1.062 ...
 $ glu  : num -1.13 2.386 -1.42 1.418 -0.453 ...
 $ bp   : num -0.285 -0.122 0.852 0.365 -0.935 ...
```

```
 $ skin : num -0.112 0.363 1.123 1.313 -0.397 ...
 $ bmi  : num -0.391 -1.132 0.423 2.181 -0.943 ...
 $ ped  : num -0.403 -0.987 -1.007 -0.708 -1.074 ...
 $ age  : num -0.708 2.173 0.315 -0.522 -0.801 ...
```

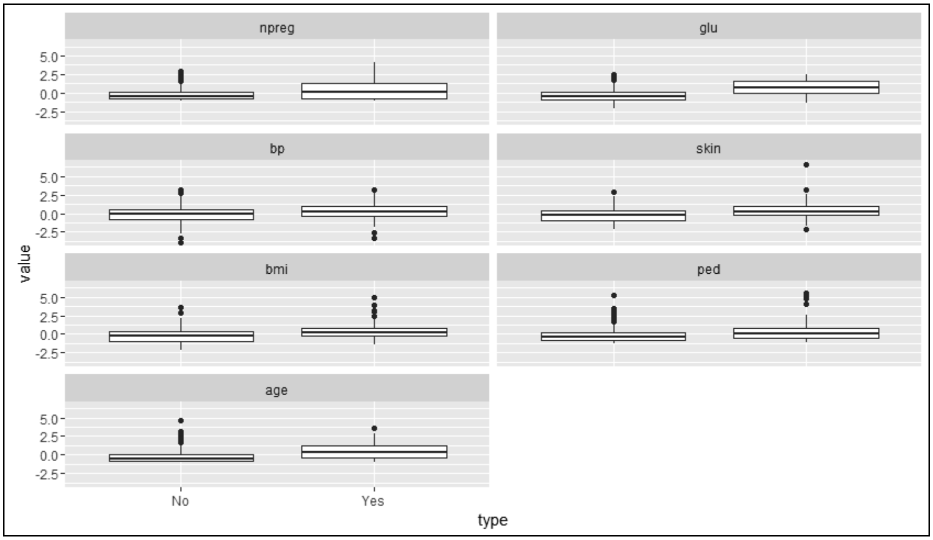
现在，我们要在数据框中加入响应变量，如下所示：

```
> pima.scale$type <- pima$type
```

使用melt()和ggplot()重新生成箱线图：

```
    > pima.scale.melt <- melt(pima.scale, id.var = "type")
> ggplot(data = pima.scale.melt, aes(x = type, y = value)) +
         geom_boxplot() + facet_wrap(~ variable, ncol = 2)
```

上述命令输出如下。



对特征进行标准化之后，箱线图好看多了。除了血糖浓度之外，可以看出其他特征也随着type发生变化，特别是age。

将数据分为训练集和测试集之前，先使用R中的cor()函数查看相关性。这个函数不会生成皮尔逊相关性统计图，而生成一个矩阵：

```
    > cor(pima.scale[-8])
            npreg       glu          bp        skin
npreg 1.000000000 0.1253296 0.204663421 0.09508511
glu   0.125329647 1.0000000 0.219177950 0.22659042
bp    0.204663421 0.2191779 1.000000000 0.22607244
skin  0.095085114 0.2265904 0.226072440 1.00000000
bmi   0.008576282 0.2470793 0.307356904 0.64742239
ped   0.007435104 0.1658174 0.008047249 0.11863557
```

```
age   0.640746866 0.2789071 0.346938723 0.16133614
              bmi       ped          age
npreg 0.008576282 0.007435104 0.64074687
glu   0.247079294 0.165817411 0.27890711
bp    0.307356904 0.008047249 0.34693872
skin  0.647422386 0.118635569 0.16133614
bmi   1.000000000 0.151107136 0.07343826
ped   0.151107136 1.000000000 0.07165413
age   0.073438257 0.071654133 1.00000000
```

有两对变量之间具有相关性：npreg/age和skin/bmi。如果能够正确训练模型，并能调整好超参数，那么多重共线性对于这些方法通常都不是问题。

我认为我们已经做好了建立训练集和测试集的准备，但是我建议一定要先检查响应变量中Yes和No的比例。确保数据划分平衡是非常重要的，如果某个结果过于稀疏，就会导致问题，可能引起分类器在优势类和劣势类之间发生偏离。对于不平衡的判定没有一个固定的规则。一个比较好的经验法则是，结果中的比例至少应该达到2：1（He与Wa，2013）。

```
    > table(pima.scale$type)
 No Yes
355 177
```

比例为2：1，现在可以建立训练集和测试集了。使用我们常用的语法，划分比例为70/30，如下所示：

```
    > set.seed(502)
> ind <- sample(2, nrow(pima.scale), replace = TRUE, prob = c(0.7,
    0.3))
> train <- pima.scale[ind == 1, ]
> test <- pima.scale[ind == 2, ]
> str(train)
'data.frame':385 obs. of 8 variables:
 $ npreg: num  0.448 0.448 -0.156 -0.76 -0.156 ...
 $ glu  : num  -1.42 -0.775 -1.227 2.322 0.676 ...
 $ bp   : num  0.852 0.365 -1.097 -1.747 0.69 ...
 $ skin : num  1.123 -0.207 0.173 -1.253 -1.348 ...
 $ bmi  : num  0.4229 0.3938 0.2049 -1.0159 -0.0712 ...
 $ ped  : num  -1.007 -0.363 -0.485 0.441 -0.879 ...
 $ age  : num  0.315 1.894 -0.615 -0.708 2.916 ...
 $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 2 1 1 1 ...
> str(test)
'data.frame':147 obs. of 8 variables:
 $ npreg: num  0.448 1.052 -1.062 -1.062 -0.458 ...
 $ glu  : num  -1.13 2.386 1.418 -0.453 0.225 ...
 $ bp   : num  -0.285 -0.122 0.365 -0.935 0.528 ...
 $ skin : num  -0.112 0.363 1.313 -0.397 0.743 ...
 $ bmi  : num  -0.391 -1.132 2.181 -0.943 1.513 ...
 $ ped  : num  -0.403 -0.987 -0.708 -1.074 2.093 ...
 $ age  : num  -0.7076 2.173 -0.5217 -0.8005 -0.0571 ...
 $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 1 2 1 1 1 ...
```

一切就绪，下一步就是建立预测模型并进行评价。我们从KNN开始。

5.3.3　模型构建与模型评价

下面讨论与模型构建和模型评价相关的各个主题。

1. KNN建模

我们在前面提到过，使用KNN建模关键的一点就是选择最合适的参数（k或K）。在确定k值方面，caret包又可以大展身手了。先建立一个供实验用的输入网格，k值从2到20，每次增加1。使用`expand.grid()`和`seq()`函数可以轻松实现。在caret包中，作用于KNN函数的参数非常简单直接，就是`.k`：

```
> grid1 <- expand.grid(.k = seq(2, 20, by = 1))
```

选择参数时，还是使用交叉验证。先建立一个名为`control`的对象，然后使用caret包中的`trainControl()`函数，如下所示：

```
> control <- trainControl(method = "cv")
```

现在，使用`train()`函数建立计算最优k值的对象，`train()`函数也在caret包中。别忘了在进行任何随机抽样之前，都要先设定随机数种子：

```
> set.seed(502)
```

使用`train()`函数建立对象时，需要指定模型公式、训练数据集名称和一个合适的方法。模型公式和以前一样——$y \sim x$，方法就是`knn`。这些参数设定之后，R代码就可以建立对象并计算最优k值了，如下所示：

```
> knn.train <- train(type ~ ., data = train,
  method = "knn",
  trControl = control,
  tuneGrid = grid1)
```

调用这个对象即可得到我们追寻的最优k值，是17：

```
    > knn.train
k-Nearest Neighbors
385 samples
  7 predictor
  2 classes: 'No', 'Yes'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 347, 347, 345, 347, 347, 346, ...
Resampling results across tuning parameters:
  k   Accuracy  Kappa  Accuracy SD  Kappa SD
   2  0.736     0.359  0.0506       0.1273
   3  0.762     0.416  0.0526       0.1313
   4  0.761     0.418  0.0521       0.1276
   5  0.759     0.411  0.0566       0.1295
   6  0.772     0.442  0.0559       0.1474
```

```
    7  0.767     0.417  0.0455       0.1227
    8  0.767     0.425  0.0436       0.1122
    9  0.772     0.435  0.0496       0.1316
   10  0.780     0.458  0.0485       0.1170
   11  0.777     0.446  0.0437       0.1120
   12  0.775     0.440  0.0547       0.1443
   13  0.782     0.456  0.0397       0.1084
   14  0.780     0.449  0.0557       0.1349
   15  0.772     0.427  0.0449       0.1061
   16  0.782     0.453  0.0403       0.0954
   17  0.795     0.485  0.0382       0.0978
   18  0.782     0.451  0.0461       0.1205
   19  0.785     0.455  0.0452       0.1197
   20  0.782     0.446  0.0451       0.1124
Accuracy was used to select the optimal model using the largest
      value.
The final value used for the model was k = 17.
```

5

除了得到k = 17这个结果之外，我们在输出的表格中还可以看到正确率和Kappa统计量的信息，以及交叉验证过程中产生的标准差。正确率告诉我们模型正确分类的百分比。Kappa又称**科恩的K统计量**，通常用于测量两个分类器对观测值分类的一致性。Kappa可以使我们对分类问题的理解更加深入，它对正确率进行了修正，去除了仅靠偶然性（或随机性）获得正确分类的因素。计算这个统计量的公式是Kappa = (一致性百分比 – 期望一致性百分比)/(1 – 期望一致性百分比)。

一致性百分比是分类器的分类结果与实际分类相符合的程度（就是正确率），**期望一致性百分比**是分类器靠随机选择获得的与实际分类相符合的程度。Kappa统计量的值越大，分类器的分类效果越好，Kappa为1时达到一致性的最大值。下面将模型应用到测试数据集上，通过一个完整的例子说明如何计算正确率和Kappa。

使用class包中的`knn()`函数实现。要使用这个函数，至少需要指定4个参数：训练数据、测试数据、训练集中的正确标记、k值。生成一个名为`knn.test`的对象，看看效果如何：

```
> knn.test <- knn(train[, -8], test[, -8], train[, 8], k = 17)
```

对象生成之后，检查混淆矩阵，算出正确率和Kappa：

```
    > table(knn.test, test$type)
knn.test No Yes
     No  77  26
     Yes 16  28
```

正确率的计算非常简单，用分类正确的观测数除以观测总数即可：

```
    > (77 + 28) / 147
[1] 0.7142857
```

正确率为71%，这比我们在训练数据上得到的正确率（约80%）稍低一些。下面看看如何使用代码计算Kappa统计量：

```
> #calculate Kappa
> prob.agree <- (77 + 28) / 147 #accuracy
> prob.chance <- ((77 + 26) / 147) * ((77 + 16) / 147)
> prob.chance
[1] 0.4432875
> kappa <- (prob.agree - prob.chance) / (1 - prob.chance)
> kappa
[1] 0.486783
```

Kappa统计量的值为0.49，和我们在训练数据集中得到的一样。Altman（1991）给出了一种启发式的方法，帮助我们解释这个统计量，如下表所示。

*k*值	一致性强度
< 0.20	很差
0.21 ~ 0.40	一般
0.41 ~ 0.60	中等
0.61 ~ 0.80	好
0.81 ~ 1.00	很好

我们的Kappa只是中等，在测试集上的正确率仅比70%高一点，所以应该看看是否可以使用加权最近邻法得到更好的结果。加权最近邻法提高了离观测更近的邻居的影响力，降低了远离观测的邻居的影响力。观测离空间点越远，对它的影响力的惩罚就越大。要使用加权最近邻法，需要kknn包中的`train.kknn()`函数来选择最优的加权方式。

`train.kknn()`函数使用我们前面介绍过的LOOCV选择最优参数，比如最优的K最近邻数量、二选一的距离测量方式，以及核函数。

在前面的讨论中，不加权的K最近邻算法使用的是欧式距离。在kknn包中，除了欧式距离，还可以选择两点坐标差的绝对值之和。如果要使用这种距离计算方式，需要指定闵可夫斯基距离参数。

有多种方法可以对距离进行加权。我们要使用的kknn包中有10种不同的加权方式，不加权也是其中之一。它们是：retangular（不加权）、triangular、epanechnikov、biweight、triweight、consine、inversion、gaussian、rank和optimal。对这些加权技术的全面介绍请参考Hechenbichler K.与Schliep K.P.（2004）。

为简单起见，我们集中讨论其中两种：triangular和epanechnikov。赋予权重之前，算法对所有距离进行标准化处理，使它们的值都在0和1之间。triangular加权方法先算出1减去距离的差，再用差作为权重去乘这个距离。epanechnikov加权方法是用3/4乘以(1 – 距离的平方)。为了方便比较，我们把这两种加权方法和标准的不加权方法放在一起实现。

先指定随机数种子，然后使用`kknn()`函数建立训练集对象，这个函数要求指定的参数有：*k*

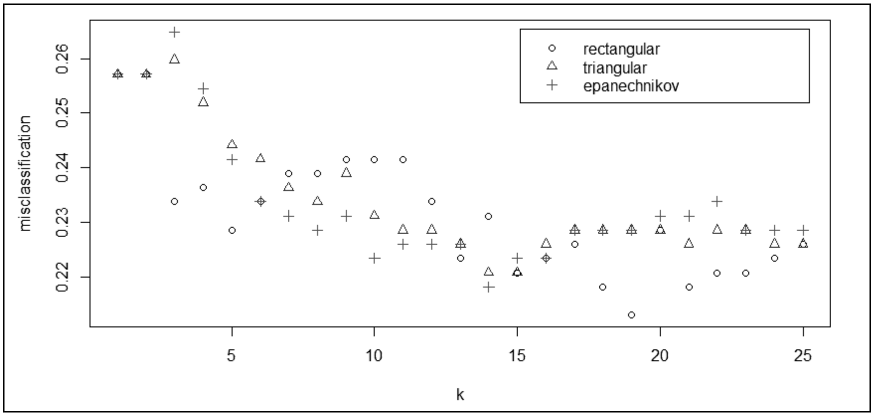
值的最大值kmax、距离distance（1表示绝对值距离，2表示欧氏距离）、核函数。在我们的模型中，kmax设定为25，distance为2：

```
> set.seed(123)
> kknn.train <- train.kknn(type ~ ., data = train, kmax = 25,
      distance = 2,
      kernel = c("rectangular", "triangular", "epanechnikov"))
```

kknn的一个特别好处是，可以用统计图的方式比较结果，如下所示：

```
> plot(kknn.train)
```

上述命令输出如下。



图中X轴表示的是k值，Y轴表示的是核函数误分类观测百分比。令人惊讶的是，不加权分类方式（rectangular）在$k = 19$的时候表现最好。可以调用对象看看分类误差和最优参数，如下所示：

```
> kknn.train
Call:
train.kknn(formula = type ~ ., data = train, kmax = 25, distance =
   2, kernel
  = c("rectangular", "triangular", "epanechnikov"))
  Type of response variable: nominal
  Minimal misclassification: 0.212987
  Best kernel: rectangular
  Best k: 19
```

从上面的数据可以看出，给距离加权不能提高模型在训练集上的正确率。而且从下面的代码可以看出，它同样不能提高测试集上的正确率：

```
> kknn.pred <- predict(kknn.train, newdata = test)
> table(kknn.pred, test$type)
kknn.pred No Yes
      No 76  27
     Yes 17  27
```

我们还可以实验其他加权方法，但其他方法的结果并不比这些更好。我们对KNN的讨论到此为止。在你自己的案例中，应该多多尝试各种参数，看看它们的效果。

2. SVM建模

使用e1071包构建SVM模型，先从线性支持向量分类器开始，然后转入非线性模型。e1071包中有一个非常好的用于SVM的函数——`tune.svm()`，它可以帮助我们选择调优参数及核函数。`tune.svm()`使用交叉验证使调优参数达到最优。我们先建立一个名为`linear.tune`的对象，然后使用`summary()`函数看看其中的内容。如下所示：

```
    > linear.tune <- tune.svm(type ~ ., data = train,
      kernel = "linear",
      cost = c(0.001, 0.01, 0.1, 1, 5, 10))
> summary(linear.tune)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 cost
    1
- best performance: 0.2051957
- Detailed performance results:
   cost     error dispersion
1 1e-03 0.3197031 0.06367203
2 1e-02 0.2080297 0.07964313
3 1e-01 0.2077598 0.07084088
4 1e+00 0.2051957 0.06933229
5 5e+00 0.2078273 0.07221619
6 1e+01 0.2078273 0.07221619
```

对于这些数据，最优成本函数`cost`是1，这时的误分类误差率差不多为21%。我们在测试集上进行预测和检验，使用`predict()`函数，指定`newdata=test`：

```
    > best.linear <- linear.tune$best.model
> tune.test <- predict(best.linear, newdata = test)
> table(tune.test, test$type)
tune.test No Yes
      No  82  22
     Yes  13  30
> (82 + 30)/147
[1] 0.7619048
```

线性支持向量分类器在训练集和测试集上表现得都比KNN稍好一些。e1071包中有一个用于SVM的非常好的函数——`tune.svm()`，它可以帮助我们选择调优参数或核函数。现在看看非线性模型能否表现得更好，依然使用交叉验证选择调优参数。

我们要试验的第一个核函数是多项式核函数，需要调整优化两个参数：多项式的阶（`degree`）与核系数（`coef0`）。设定多项式的阶是3、4和5，核系数从0.1逐渐增加到4，如下所示：

```
> set.seed(123)
> poly.tune <- tune.svm(type ~ ., data = train,
      kernel = "polynomial",
      degree = c(3, 4, 5),
      coef0 = c(0.1, 0.5, 1, 2, 3, 4))
> summary(poly.tune)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 degree coef0
      3   0.1
- best performance: 0.2310391
```

模型选择的多项式阶数为3，核系数为0.1。和线性SVM一样，可以用这些参数在测试集上进行预测，如下所示：


```
> best.poly <- poly.tune$best.model
> poly.test <- predict(best.poly, newdata = test)
> table(poly.test, test$type)
poly.test No Yes
      No  81  28
     Yes  12  26
> (81 + 26) / 147
[1] 0.7278912
```

这个模型的表现还不如线性模型。下面测试径向基核函数，此处只需找出一个参数gamma，在0.1 ~ 4中依次检验。如果gamma过小，模型就不能解释决策边界的复杂性；如果gamma过大，模型就会严重过拟合。

```
> set.seed(123)
> rbf.tune <- tune.svm(type ~ ., data = train,
      kernel = "radial",
      gamma = c(0.1, 0.5, 1, 2, 3, 4))
> summary(rbf.tune)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma
   0.5
- best performance: 0.2284076
```

最优的gamma值是0.5，这种模型的表现看上去也不比其他SVM模型好多少。我们还是要在测试集上进行验证，和前面一样：

```
> best.rbf <- rbf.tune$best.model
> rbf.test <- predict(best.rbf, newdata = test)
> table(rbf.test, test$type)
rbf.test No Yes
     No  73  33
    Yes  20  21
> (73+21)/147
[1] 0.6394558
```

表现可谓惨不忍睹。提高性能的最后一招只能是kernel="sigmoid"了。需要找出两个参数——gamma和核系数（coef0）：

```
> set.seed(123)
> sigmoid.tune <- tune.svm(type ~ ., data = train,
      kernel = "sigmoid",
      gamma = c(0.1, 0.5, 1, 2, 3, 4),
      coef0 = c(0.1, 0.5, 1, 2, 3, 4))
> summary(sigmoid.tune)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma coef0
   0.1     2
- best performance: 0.2080972
```

误差率和线性模型基本一样。现在只要看看能否在测试集上表现得更好：

```
> best.sigmoid <- sigmoid.tune$best.model
> sigmoid.test <- predict(best.sigmoid, newdata = test)
> table(sigmoid.test, test$type)
sigmoid.test No Yes
         No  82  19
         Yes 11  35
> (82+35)/147
[1] 0.7959184
```

看看吧！我们终于找到了一个在测试集上和训练集上有相同表现的模型了。看起来，可以选择sigmoid核函数作为最优预测。

迄今为止，我们一直在折腾各种不同类型的模型。下面要对这些模型（包括线性模型）的性能进行评价，不只看正确率，而要使用各种指标。

5.3.4　模型选择

我们已经研究了两种不同类型的建模技术，从各方面来看，KNN都处于下风。KNN在测试集上最好的正确率只有71%左右，相反，通过SVM可以获得接近80%的正确率。在简单选择具有最优正确率的模型（本案例中是使用sigmoid核函数的SVM模型）之前，先看看如何通过对混淆矩阵的深入研究来比较各种模型。

为了完成这个任务，还是要求助于我们的老朋友caret包，使用其中的confusionMatrix()函数。请注意，在前面的内容中，我们使用过InformationValue包中的同名函数。但caret包中的这个函数会生成我们评价和选择最优模型所需的所有统计量。先从建立的最后一个模型开始，使用的语法和基础的table()函数一样，不同之处是要指定positive类，如下所示：

```
> confusionMatrix(sigmoid.test, test$type, positive = "Yes")
Confusion Matrix and Statistics
```

```
          Reference
Prediction No Yes
       No  82  19
       Yes 11  35
               Accuracy : 0.7959
                 95% CI : (0.7217, 0.8579)
    No Information Rate : 0.6327
    P-Value [Acc > NIR] : 1.393e-05
                  Kappa : 0.5469
 Mcnemar's Test P-Value : 0.2012
            Sensitivity : 0.6481
            Specificity : 0.8817
         Pos Pred Value : 0.7609
         Neg Pred Value : 0.8119
             Prevalence : 0.3673
         Detection Rate : 0.2381
   Detection Prevalence : 0.3129
      Balanced Accuracy : 0.7649
       'Positive' Class : Yes
```


这个函数生成的一些项目我们已经介绍过了，比如正确率和Kappa，其他统计量介绍如下。

- No Information Rate：最大分类所占的比例——63%的人没有糖尿病。
- P-Value：用来进行假设检验，说明正确率确实高于No Information Rate。
- Mcnemar's Test：我们现在不关心这个统计量，它用于配对分析，主要用于流行病学的研究。
- Sensitivity：敏感度，真阳性率；在本案例中，表示没有糖尿病并且被正确识别的比例。
- Specificity：特异度，真阴性率；在本案例中，表示有糖尿病并且被正确识别的比例。
- Pos Pred Value：阳性预测率，被认为有糖尿病的人中真的有糖尿病的概率。使用下面的公式计算。

$$PPV = \frac{敏感度 * 患病率}{(敏感度 * 患病率) + (1 - 敏感度) * (1 - 患病率)}$$

- Neg Pred Value：阴性预测率，被认为没有糖尿病的人中真的没有糖尿病的概率。使用下面的公式计算。

$$NPV = \frac{敏感度 * (1 - 患病率)}{((1 - 敏感度) * (患病率)) + (敏感度) * (1 - 患病率)}$$

- Prevalence：患病率，某种疾病在人群中流行度的估计值，本例中的计算方法是第二列（Yes列）中的数之和除以总观测数（矩阵中所有数之和）。
- Detection Rate：真阳性预测中被正确识别的比例，在本案例中，用35除以总观测数。
- Detection Prevalence：预测的患病率，在本案例中，底行中的数的和除以总观测数。

- Balanced Accuracy：所有类别正确率的平均数。用来表示由于分类器算法中潜在的偏差造成的对最频繁类的过度预测。可以简单地用(敏感度 + 特异度)/2来计算。

我们的模型敏感度不像希望的那么高，这说明从数据集中丢失了一些特征，这些特征可以提高找出真正糖尿病患者的可能性。我们用这些结果和线性SVM进行对比，如下所示：

```
    > confusionMatrix(tune.test, test$type, positive = "Yes")
          Reference
Prediction No Yes
       No  82  24
       Yes 11  30
               Accuracy : 0.7619
                 95% CI : (0.6847, 0.8282)
    No Information Rate : 0.6327
    P-Value [Acc > NIR] : 0.0005615
                  Kappa : 0.4605
 Mcnemar's Test P-Value : 0.0425225
            Sensitivity : 0.5556
            Specificity : 0.8817
         Pos Pred Value : 0.7317
         Neg Pred Value : 0.7736
             Prevalence : 0.3673
         Detection Rate : 0.2041
   Detection Prevalence : 0.2789
      Balanced Accuracy : 0.7186
       'Positive' Class : Yes
```

从两个模型的对比可以看出，线性SMV模型在各方面都处于下风，sigmoid核函数模型明显更胜一筹。但此处忽略了一件事，即没有进行任何特征选择。我们做的工作就是把特征堆在一起，作为所谓的输入空间，然后让SVM这个黑盒去计算，最后给出一个预测分类。使用SVM的一个主要问题就是，它给出的结果非常难以解释。有一些方法可以改进这个问题，但我认为超出了本章范围。完全掌握了前面讲述的基础知识之后，这些事情应该由你独立探索和研究。

5.4　SVM 中的特征选择

但是，还有一些其他办法可以进行特征选择，我要花一些篇幅来告诉你们开始这项工作的捷径，你需要做的就是反复实验。我们要再次用到caret包，因为它可以基于kernlab包在线性SVM中进行交叉验证。

要完成上面的任务，需要设定随机数种子，在caret包中的`rfeControl()`函数中指定交叉验证方法，使用`rfe()`函数执行一个递归的特征选择过程，最后检验模型在测试集上的运行情况。在`rfeControl()`中，你需要根据使用的模型指定`functions`参数。可以使用几种不同的`functions`参数，此处使用`lrFuncs`。如果想看可用的`functions`参数列表，最好使用`?rfeControl`和`?caretFuncs`命令查看相关文档。本例代码如下：

```
    > set.seed(123)
> rfeCNTL <- rfeControl(functions = lrFuncs, method = "cv", number
    = 10)
    > svm.features <- rfe(train[, 1:7], train[, 8],
      sizes = c(7, 6, 5, 4),
      rfeControl = rfeCNTL,
      method = "svmLinear")
```

要建立`svm.features`对象，重点是要指定输入数据和响应因子、通过参数`sizes`指定输入特征的数量以及kernlab包中的线性方法（此处是`svmLinear`）。method还有其他一些选项，比如`svmPoly`，但是没有对应于sigmoid核函数的选项。调用这个对象即可查看具有不同数量特征的模型的表现，如下所示：

```
    > svm.features
Recursive feature selection
Outer resampling method: Cross-Validated (10 fold)
Resampling performance over subset size:
 Variables Accuracy Kappa AccuracySD KappaSD Selected
         4 0.7797  0.4700    0.04969  0.1203
         5 0.7875  0.4865    0.04267  0.1096        *
         6 0.7847  0.4820    0.04760  0.1141
         7 0.7822  0.4768    0.05065  0.1232
The top 5 variables (out of 5):
```

出乎意料，五特征模型表现得相当好，和包括`skin`和`bp`的全特征模型不相上下。我们在测试集上进行实验，别忘了，全特征模型的正确率是76.2%：

```
    > svm.5 <- svm(type ~ glu + ped + npreg + bmi + age,
      data = train,
      kernel = "linear")
> svm.5.predict <- predict(svm.5, newdata = test[c(1, 2, 5, 6, 7)])
> table(svm.5.predict, test$type)
svm.5.predict No Yes
          No  79  21
          Yes 14  33
```

表现不怎么样，我们还是要回到全特征模型。通过反复实验，你可以看到这种方法是如何工作的，它的目的是对特征重要性做一个简单的鉴别。如果你想研究其他可以用于特征选择的技术和方法（特别是那些用于黑盒技术的），我推荐你先了解Guyon和Elisseeff（2003）在这个领域的工作。

5.5 小结

我们在本章讨论了两种新的分类技术：KNN和SVM，并在一个小规模数据集上分别使用这两种技术建立了模型，以预测某个人是否患有糖尿病，还对模型进行了比较。我们的目的是说明KNN和SVM的工作原理及其区别。在KNN部分，我们介绍了不加权和加权的最近邻算法。在预测某个人是否患有糖尿病方面，这些方法的表现都不如SVM。

我们研究了如何使用e1701包建立线性的和非线性的支持向量机，并对其进行调优。使用功能极其强大而又全面的caret包比较线性和非线性支持向量机的预测能力，然后发现，使用sigmoid核函数的非线性支持向量机具有最好的性能。

最后，我们简单介绍了如何使用caret包进行粗略的特征选择。因为对于那些像SVM一样使用黑盒技术的方法来说，特征选择确实是个艰巨的挑战。这也是使用这些技术时可能遇到的主要困难，在解决商业问题时，你一定要仔细斟酌这些技术的可行性。

第 6 章

分类回归树

“最好的分类器可能是随机森林（RF），最好的版本（用R开发，通过caret调用）可以在84.3%的数据集上达到94.1%的最大正确率，超过了90%的其他分类器。”

——费尔南德斯–德尔加多 等（2014）

上文引自费尔南德斯–德尔加多等人在《机器学习研究期刊》上发表的一篇论文，从这段话可以看出，我们在这一章要讨论的技术非常强大，特别是用于分类问题时。当然，它们不是永远的最优解，但确实提供了一个非常好的起点。

我们之前研究的技术或者用于预测定量结果，或者用于预测分类问题。本章要讨论的技术对这两类问题都适用，而处理业务问题的方式和前几章不同。我们不引入新的问题，而是把这些技术用于前面已经解决的问题，着眼于这些技术能否提高预测能力。总而言之，本章案例的目的就是要看看是否能继续改善以前的模型。

第一个要讨论的项目是基础的决策树，模型建立和解释都非常简单。但单决策树模型的表现并不如我们研究过的模型那样好，比如支持向量机模型；也比不上我们将要学习的模型，比如神经网络模型。所以，我们要讨论的是建立多个（有时是上百个）不同的树，然后把每个树的结果组合在一起，最后形成一个综合的预测。

正像本章开头引用的论文所说，这些方法产生的效果不次于甚至超过书中涉及的所有其他技术。这些技术被称为**随机森林**和**梯度提升树**。此外，在讨论商业案例之余，我们还会介绍如何在数据集上应用随机森林方法，以帮助实现特征消除（特征选择）。

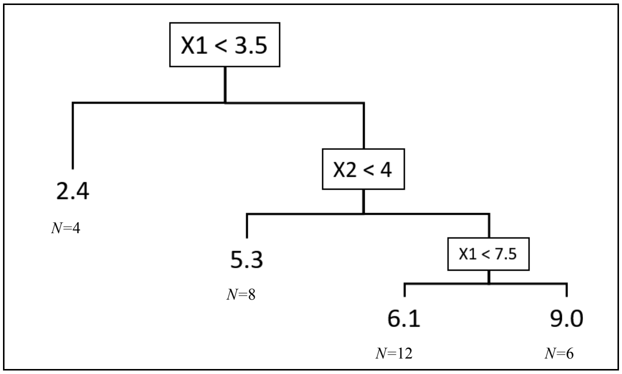
6.1 本章技术概述

我们要简单介绍本章涵盖的技术，包括回归树与分类树、随机森林和梯度提升，为使用这些技术分析实际商业案例打下基础。

6.1.1　回归树

要想理解基于树的方法，比较容易的方式是从预测定量结果开始，然后学习它是如何解决分类问题的。树方法的精髓就是划分特征，从第一次分裂开始就要考虑如何最大程度改善RSS，然后持续进行二叉分裂，直到树结束。后面的划分并不作用于全体数据集，而仅作用于上次划分时落到这个分支之下的那部分数据。这个自顶向下的过程被称为"递归划分"。这个过程是**贪婪**的，这个名词在研究机器学习方法时会经常遇到。贪婪的含义是，算法在每次分裂中都追求最大程度减少RSS，而不管以后的划分中表现如何。这样做的结果是，你可能会生成一个带有无效分支的树，尽管偏差很小，但是方差很大。为了避免这个问题，生成完整的树之后，你要对树进行剪枝，得到最优的规模。

下图给出了使用树进行分类的一个直观的实际例子。假设数据集中有30个观测，响应变量的范围是1 ~ 10，有两个预测特征X_1和X_2，范围都是0 ~ 10。整个树有3个分支和4个终端节点。每次都按照基本的`if-then`语句，或者使用R函数`ifelse()`进行分裂。第一次分裂的条件是X_1是否小于3.5，响应变量被分出4个观测，平均值为2.4，剩余26个观测。包含4个观测的左侧分支是一个终端节点，因为更深的分裂已经不能明显改善RSS了。树的这部分划分包含的4个观测的预测值就是平均值2.4。下一次分裂的条件是$X_2 < 4$，最后是$X_1 < 7.5$。



有3个分支和4个节点的回归树，节点下面标有均值和观测数量

这种方法的优点是可以处理高度非线性关系，但它还存在一些潜在的问题，你能发现吗？首要的问题就是，一个观测被赋予所属终端节点的平均值，这会损害整体预测效果（高偏差）。相反，如果你一直对数据进行划分，树的层次越来越深，这样可以达到低偏差的效果，但是高方差又成了问题。和其他方法一样，你也可以用交叉验证来选择合适的深度。

6.1.2　分类树

分类树与回归树的运行原理是一样的，区别在于决定分裂过程的不是RSS，而是误差率。这

里的误差率不是你想的那样，简单地由误分类的观测数除以总观测数算出。实际上，进行树分裂时，误分类率本身可能会导致这样一种情况：你可以从下次分裂中获得一些有用信息，但误分类率却没有改善。我们看一个例子。

假设有一个节点N_0，节点中有7个标号为`No`的观测和3个标号为`Yes`的观测，我们就可以说误分类率为30%。记住这个数，然后通过另一种误差测量方式进行计算，这种方式称为**基尼指数**。单个节点的基尼指数计算公式如下：

$$基尼指数 = 1 - (类别1的概率)^2 - (类别2的概率)^2$$

所以，对于N_0，基尼指数为$1 - (0.7)^2 - (0.3)^2$，等于0.42，与之相对的误分类率为30%。

将这个例子讨论得更深入一些，假设将节点N_0分裂成两个节点N_1和N_2，N_1中有3个观测属于类别1，没有属于类别2的观测；N_2中有4个观测属于类别1，3个属于类别2。现在，树的这个分支的整体的误分类率还是30%，那么看看整体的基尼指数是如何改善的：

- 基尼指数$(N_1) = 1 - (3/3)^2 - (0/3)^2 = 0$
- 基尼指数$(N_2) = 1 - (4/7)^2 - (3/7)^2 = 0.49$
- 新基尼指数 = (N_1比例 × 基尼指数(N_1))+(N_2比例 × 基尼指数(N_2)) = (0.3 × 0) + (0.7 × 0.49)=0.343

通过一次基于替代误差率的分裂，我们确实改善了模型的不纯度，将其从原来的0.42减小到0.343，误分类率却没有变化。`rpart()`包就是使用Gini指数测量误差的，在本章中我们会使用这个程序包。

6.1.3　随机森林

为了显著提高模型的预测能力，我们可以生成多个树，然后将这些树的结果组合起来。随机森林技术在模型构建过程中使用两种奇妙的方法，以实现这个构想。第一个方法称为**自助聚集**，或称**装袋**。

在装袋法中，使用数据集的一次随机抽样建立一个独立树，抽样的数量大概为全部观测的2/3（请记住，剩下的1/3被称为**袋外数据**，out-of-bag）。这个过程重复几十次或上百次，最后取平均结果。其中每个树都任其生长，不进行任何基于误差测量的剪枝，这意味着每个独立树的方差都很大。但是，通过对结果的平均化处理可以降低方差，同时又不增加偏差。

另一个在随机森林中使用的方法是，对数据进行随机抽样（装袋）的同时，独立树每次分裂时对输入特征也进行随机抽样。在randomForest包中，我们使用随机抽样数的默认值来对预测特征进行抽样。对于分类问题，默认值为所有预测特征数量的平方根；对于回归问题，默认值为所有预测特征数量除以3。在模型调优过程中，每次树分裂时，算法随机选择的预测特征数量是可变的。

通过每次分裂时对特征的随机抽样以及由此形成的一套方法，你可以减轻高度相关的预测特征的影响，这种预测特征在由装袋法生成的独立树中往往起主要作用。这种方法还使你不必思索如何减少装袋导致的高方差。独立树彼此之间的相关性减少之后，对结果的平均化可以使泛化效果更好，对于异常值的影响也更加不敏感，比仅进行装袋的效果要好。

6.1.4　梯度提升

提升法的学习和理解可能会非常复杂，但你一定要对其中原理有基本的了解。它的主要思想是，先建立一个某种形式的初始模型（线性、样条、树或其他），称为基学习器；然后检查残差，在残差的基础上围绕**损失函数**拟合模型。损失函数测量模型和现实之间的差别，例如，在回归问题中可以用误差的平方，在分类问题中可以用逻辑斯蒂函数。一直继续这个过程，直到满足某个特定的结束条件。这与下面的情形有点相似：一个学生进行模拟考试，100道题中错了30道，然后只研究那30道错题；在下次模考中，30道题中又错了10道，然后只研究那10道题，以此类推。如果你想更加深入地研究背后的理论，可以参考Frontiers in Neurorobotics, *Gradient boosting machines, a tutorial*, Natekin A., Knoll A. (2013)（http://www.ncbi.nlm.nih.gov/pmc/atricles/PMC3885826/）。

刚才提到过，提升方法可以用于很多不同类型的基分类器，本章只集中讨论**基于树的学习**这个主题。 每个树迭代的次数都很少，我们要通过一个调优参数决定树的分裂次数，这个参数称为交互深度。事实上，有些树很小，只可以分裂一次，这样的树就被称为“树桩”。

这些树依次按照损失函数拟合残差，直至达到我们指定的树的数量（我们的结束条件）。

使用Xgboost包进行建模的过程中，有一些参数需要调整。Xgboost表示**eXtreme Gradient Boosting**。这个程序包表现优异，广泛应用于网上的数据科学竞赛。在下面这个网站中，你可以找到关于提升树和Xgboost的非常详尽的背景资料：

http://xgboost.readthedocs.io/en/latest/model.html

在后面的商业案例中，我们要看看如何找到最优的超参数，并使模型生成有意义的结果和预测。这些参数之间会互相作用，如果你只攻一点而不顾其他，模型的效果可能会越来越差。在调参过程中，caret包会助你一臂之力。

6.2　商业案例

在前几章中，我们通过努力建立了一些模型，本章的总体目标是看看我们能否提高这些模型的预测能力。对于回归问题，重新回到第4章中的前列腺癌数据集，以0.444的均方误差作为基准，看看能否改善。

对于分类问题，我们使用第3章中的乳腺癌数据集和第5章中的皮玛印第安人糖尿病数据集。

在乳腺癌数据集中，我们取得了97.6%的预测正确率。在糖尿病数据集中，正确率是79.6%，我们希望再提高一些。

随机森林和提升方法都会用于这3个数据集，简单树模型只用于乳腺癌数据集和前列腺癌数据集。

6.2.1　模型构建与模型评价

要进行模型构建，需要加载7个不同的R包。我们会实验每种技术，并与前面章节中介绍的方法进行对比，看看它们在数据分析中会有什么突出表现。

1. 回归树

我们直接跳到前列腺癌数据集，但首先要加载所需的R包。当然，要确定在加载之前你已经安装了它们：

```
  >   library(rpart) #classification and regression trees
> library(partykit) #treeplots
> library(MASS) #breast and pima indian data
> library(ElemStatLearn) #prostate data
> library(randomForest) #random forests
> library(xgboost) #gradient boosting
> library(caret) #tune hyper-parameters
```

6

先用prostate数据集做回归，像第4章中那样准备好数据，包括调出数据集。使用`ifelse()`函数将gleason评分编码为指标变量，划分训练数据集和测试数据集，训练数据集为pros.train，测试数据集为pros.test，如下所示：

```
  >   data(prostate)
> prostate$gleason <- ifelse(prostate$gleason == 6, 0, 1)
> pros.train <- subset(prostate, train == TRUE)[, 1:9]
> pros.test <- subset(prostate, train == FALSE)[, 1:9]
```

要在训练数据集上建立回归树，需要使用party包中的`rpart()`函数，语法与我们在其他建模技术中使用过的非常类似：

```
> tree.pros <- rpart(lpsa ~ ., data = pros.train)
```

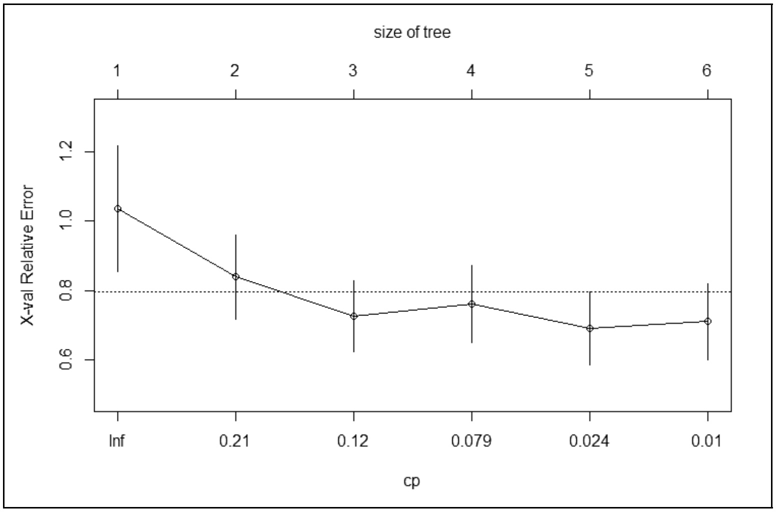
可以调出这个对象，然后检查每次分裂的误差，以决定最优的树分裂次数：

```
  > print(tree.pros$cptable)
   CP nsplit rel error  xerror   xstd
1 0.35852251    0 1.0000000 1.0364016 0.1822698
2 0.12295687    1 0.6414775 0.8395071 0.1214181
3 0.11639953    2 0.5185206 0.7255295 0.1015424
4 0.05350873    3 0.4021211 0.7608289 0.1109777
5 0.01032838    4 0.3486124 0.6911426 0.1061507
6 0.01000000    5 0.3382840 0.7102030 0.1093327
```

这个表对于我们的分析非常重要。标号为CP的第一列是成本复杂性参数，第二列nsplit是树分裂的次数，rel error列表示相对误差，即某次分裂的RSS除以不分裂的RSS（RSS(*k*)/RSS(0)），xerror和xstd都是基于10折交叉验证的，xerror是平均误差，xstd是交叉验证过程的标准差。可以看出，5次分裂在整个数据集上产生的误差最小，但使用交叉验证时，4次分裂产生的误差略微更小。可以使用`plotcp()`函数查看统计图：

```
> plotcp(tree.pros)
```

上述命令输出如下。



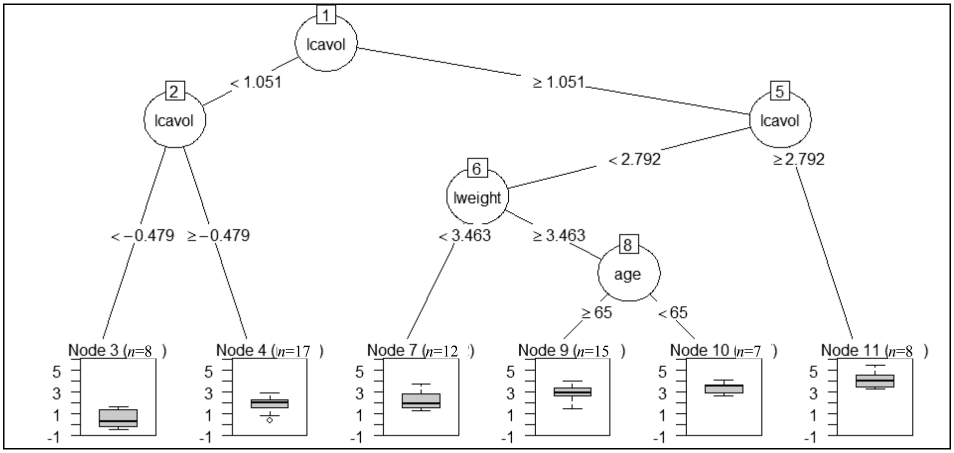
这幅图使用误差条表示树的规模和相对误差之间的关系，误差条和树规模是对应的。选择树的规模为5，也就是经过4次分裂可以建立一个新的树对象，这个树的xerror可以通过剪枝达到最小化。剪枝的方法是先建立一个cp对象，将这个对象和表中第5行相关联，然后使用`prune()`函数完成剩下的工作：

```
  >   cp <- min(tree.pros$cptable[5, ])
> prune.tree.pros <- prune(tree.pros, cp = cp)
```

完成上面的代码后，可以用统计图比较完整树和剪枝树。由partykit包生成的树图明显优于party包生成的，在`plot()`函数中，你可以简单地使用`as.party()`函数作为包装器函数：

```
> plot(as.party(tree.pros))
```

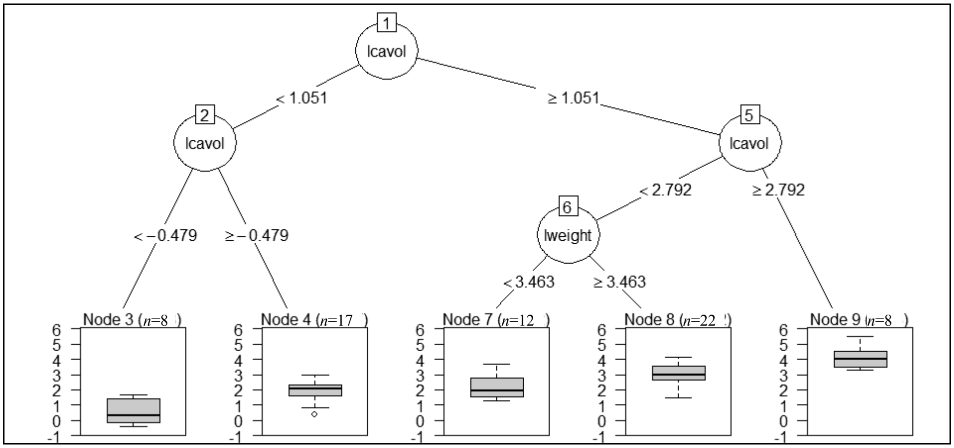
上述命令输出如下页图。



同样地，使用as.party()函数处理剪枝树：

```
> plot(as.party(prune.tree.pros))
```

上述命令输出如下。



请注意，除了最后一次分裂（完整树包含变量age），两个树是完全一样的。有趣的是，树的第一次分裂和第二次分裂都与肿瘤大小的对数（lcavol）有关。这两张图的信息量都很大，因为它们显示了树的分裂、节点、每节点观测数，以及预测结果的箱线图。

我们看看剪枝树在测试集上表现如何。在测试数据上使用predict()函数进行预测，并建立一个对象保存这些预测值。然后用预测值减去实际值，得到误差，最后算出误差平方的平均值：

```
  > party.pros.test <- predict(prune.tree.pros, newdata = pros.test)
> rpart.resid <- party.pros.test - pros.test$lpsa
> mean(rpart.resid^2) #caluclate MSE
[1] 0.5267748
```

第4章中的基准MSE为0.44，我们的努力并没有改善预测结果。但是，这种方法并非一无是处。从生成的树图中可以看出对响应变量影响最大的特征，并且很容易解释。就像引言中说过的，树模型很容易理解和解释，这往往比正确率重要得多。

2. 分类树

对于分类问题，我们先像第3章一样准备好乳腺癌数据。加载数据之后，你要删除患者ID，对特征进行重新命名，删除一些缺失值，然后建立训练数据集和测试数据集。如下所示：

```
  > data(biopsy)
> biopsy <- biopsy[, -1] #delete ID
> names(biopsy) <- c("thick", "u.size", "u.shape", "adhsn",
    "s.size", "nucl",
  "chrom", "n.nuc", "mit", "class") #change the feature names
> biopsy.v2 <- na.omit(biopsy) #delete the observations with
    missing values
> set.seed(123) #random number generator
> ind <- sample(2, nrow(biopsy.v2), replace = TRUE, prob = c(0.7,
    0.3))
> biop.train <- biopsy.v2[ind == 1, ] #the training data set
> biop.test <- biopsy.v2[ind == 2, ] #the test data set
```

准备好合适的数据之后，使用和前面回归问题一样的代码风格来解决分类问题。建立分类树之前，要确保结果变量是一个因子，可以用`str()`函数进行检查：

```
 > str(biop.test[, 10])
Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 2 1 1 ...
```

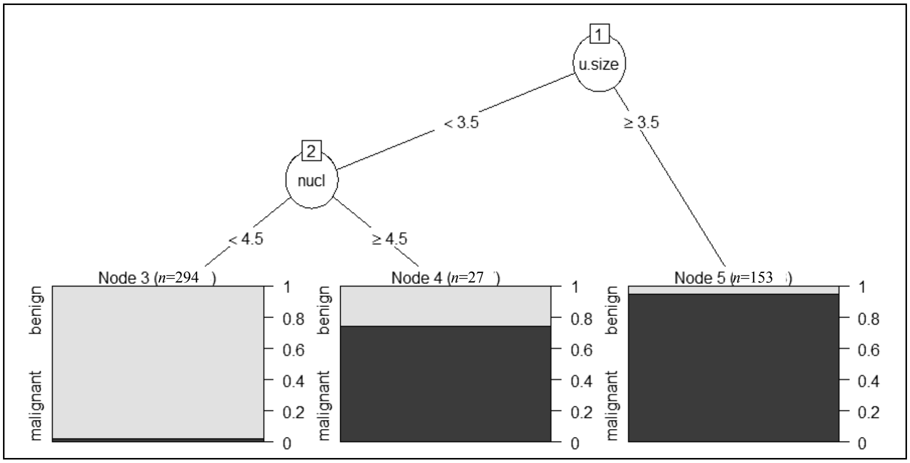
先生成树，然后检查输出中的表格，找到最优分裂次数：

```
    > set.seed(123)
> tree.biop <- rpart(class ~ ., data = biop.train)
> tree.biop$cptable
  CP nsplit rel error  xerror    xstd
1 0.79651163    0 1.0000000 1.0000000 0.06086254
2 0.07558140    1 0.2034884 0.2674419 0.03746996
3 0.01162791    2 0.1279070 0.1453488 0.02829278
4 0.01000000    3 0.1162791 0.1744186 0.03082013
```

交叉验证误差仅在两次分裂后就达到了最小值（见第3行）。现在可以对树进行剪枝，再在图中绘制剪枝树，看看它在测试集上的表现：

```
  > cp <- min(tree.biop$cptable[3, ])
> prune.tree.biop <- prune(tree.biop, cp = cp)
> plot(as.party(prune.tree.biop))
```

上述命令输出如下页图。



从对树图的检查中可以发现，细胞大小均匀度是第一个分裂点，第二个是nuclei。完整树还有一个分支是细胞浓度。使用`predict()`函数并指定`type="class"`，在测试集上进行预测。如下所示：


```
  > rparty.test <- predict(prune.tree.biop, newdata = biop.test, type
   ="class")
> table(rparty.test, biop.test$class)
rparty.test benign malignant
 benign    136     3
 malignant   6    64
> (136+64)/209
[1] 0.9569378
```

只有两个分支的基本树模型给出了差不多96%的正确率。这虽然比不上逻辑斯蒂回归的正确率97.6%，但足以激励我们相信，在随后的随机森林方法中会得到更好的改善效果。

3. 随机森林回归

在这一节中，我们还是先处理前列腺癌数据集，然后处理乳腺癌数据集和皮玛印第安人糖尿病数据集，使用的程序包是randomForest。建立一个随机森林对象的通用语法是使用`randomForest()`函数，指定模型公式和数据集这两个基本参数。回想一下每次树迭代默认的变量抽样数，对于回归问题，是$p/3$；对于分类问题，是p的平方根，p为数据集中预测变量的个数。对于大规模数据集，就p而言，你可以调整`mtry`参数，它可以确定每次迭代的变量抽样数值。如果p小于10，可以省略上面的调整过程。想在多特征数据集中优化`mtry`参数时，可以使用caret包，或使用randomForest包中的`tuneRF()`函数。了解这些情况之后，即可建立随机森林模型并检查模型结果。如下所示：

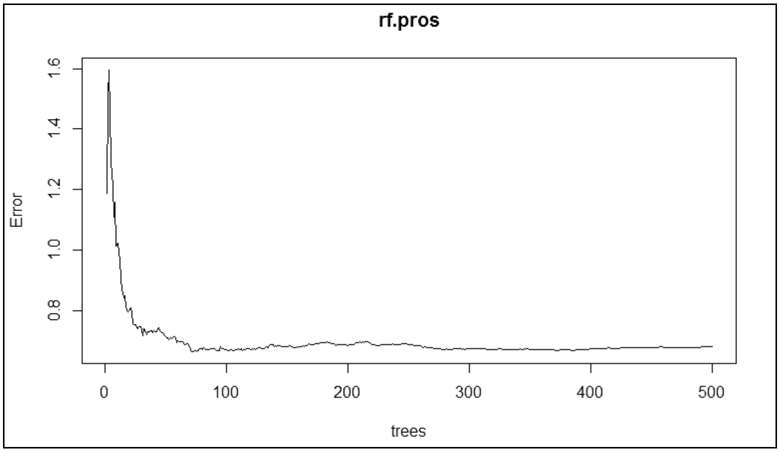
```
  > set.seed(123)
> rf.pros <- randomForest(lpsa ~ ., data = pros.train)
> rf.pros
```

```
Call:
  randomForest(formula = lpsa ~ ., data = pros.train)
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 2
  Mean of squared residuals: 0.6792314
  % Var explained: 52.73
```

调用rf.pros对象，可以知道随机森林生成了500个不同的树（默认设置），并且在每次树分裂时随机抽出两个变量。结果的MSE为0.68，差不多53%的方差得到了解释。下面看看能否对默认数量的树做一些改善。过多的树会导致过拟合，当然，“多大的数量是‘过多’”依赖于数据规模。两件事可以帮助我们达到目的，第一是做出rf.pros的统计图，另一件是求出最小的MSE：

```
> plot(rf.pros)
```

上述命令输出如下。



这个图表示MSE与模型中树的数量之间的关系。可以看出，树的数量增加时，一开始MSE会有显著改善，当森林中大约建立了100棵树之后，改善几乎停滞。

可以通过`which.min()`函数找出具体的最优树数量，如下所示：

```
  > which.min(rf.pros$mse)
[1] 75
```

可以试试有75棵树的随机森林，在模型语法中指定`ntree = 75`即可：

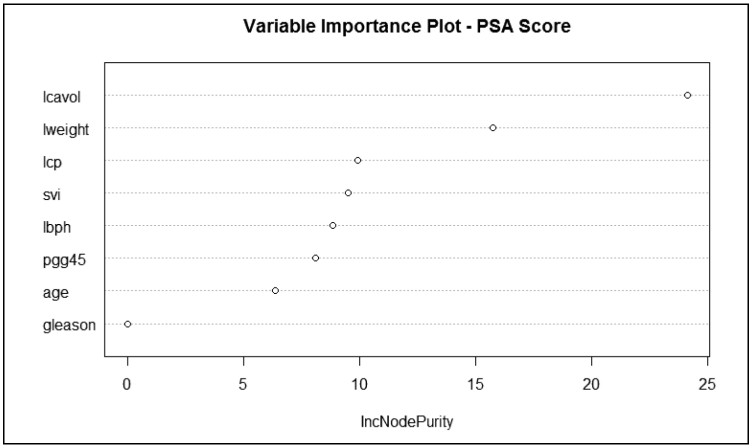
```
  > set.seed(123)
> rf.pros.2 <- randomForest(lpsa ~ ., data = pros.train, ntree
   =75)
> rf.pros.2
Call:
   randomForest(formula = lpsa ~ ., data = pros.train, ntree = 75)
   Type of random forest: regression
```

```
Number of trees: 75
No. of variables tried at each split: 2
Mean of squared residuals: 0.6632513
% Var explained: 53.85
```

可以看到，MSE和解释方差都有一点微小的改善。对模型进行检验之前，先看看另一张统计图。如果使用自助抽样和两个随机预测变量建立了75棵不同的树，要想将树的结果组合起来，需要一种方法确定哪些变量驱动着结果。独木不成林，你需要做出变量重要性统计图及相应的列表。Y轴是按重要性降序排列的变量列表，X轴是MSE改善百分比。请注意，在分类问题中，X轴应该是基尼指数的改善。使用varImpPlot()函数生成统计图：

```
> varImpPlot(rf.pros.2, scale = T,
 main = "Variable Importance Plot - PSA Score")
```

上述命令输出如下。



和单个树模型一致，lcavol是最重要的变量，lweight次之。如果想查看具体数据，可以使用importance()函数。如下所示：

```
  > importance(rf.pros.2)
    IncNodePurity
lcavol   24.108641
   lweight  15.721079
     age  6.363778
     lbph  8.842343
     svi  9.501436
     lcp  9.900339
   gleason  0.000000
    pgg45  8.088635
```

现在可以看看模型在测试数据上的表现：

```
  > rf.pros.test <- predict(rf.pros.2, newdata = pros.test)
> rf.resid = rf.pros.test - pros.test$lpsa #calculate residual
```

```
> mean(rf.resid^2)
[1] 0.5136894
```

MSE依然高于我们在第4章中使用LASSO得到的0.44，也不比单个树模型更好。

4. 随机森林分类

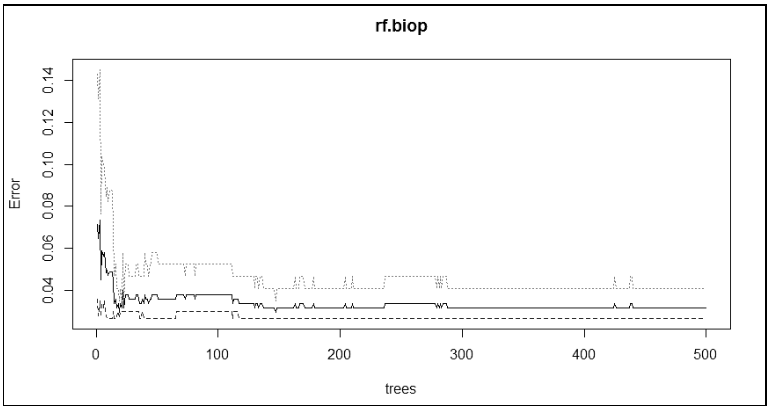
你可能会对随机森林回归模型的表现感到失望，但这种技术的真正威力在于解决分类问题。我们从乳腺癌诊断数据开始，这个过程和在回归问题中的做法几乎一样：

```
  > set.seed(123)
> rf.biop <- randomForest(class ~. , data = biop.train)
> rf.biop
Call:
 randomForest(formula = class ~ ., data = biop.train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3
        OOB estimate of error rate: 3.16%
Confusion matrix:
          benign malignant class.error
benign       294        8 0.02649007
malignant      7      165 0.04069767
```

OOB（袋外数据）误差率为3.16%。随机森林还是由默认的全部500棵树组成。画出表示误差和树的数量关系的统计图：

```
> plot(rf.biop)
```

上述命令输出如下。



图中显示，误差和标准误差的最小值是在树的数量很少的时候取得的，可以使用`which.min()`函数找出具体值。和前面不同的一点是，需要指定第一列来得到误差率，这是整体误差率。算法会为每个类标号的误差率生成一个附加列，本例中不需要它们。模型结果中也没有`mse`，而是代之以`err.rate`。如下页代码所示：

```
> which.min(rf.biop$err.rate[, 1])
[1] 19
```

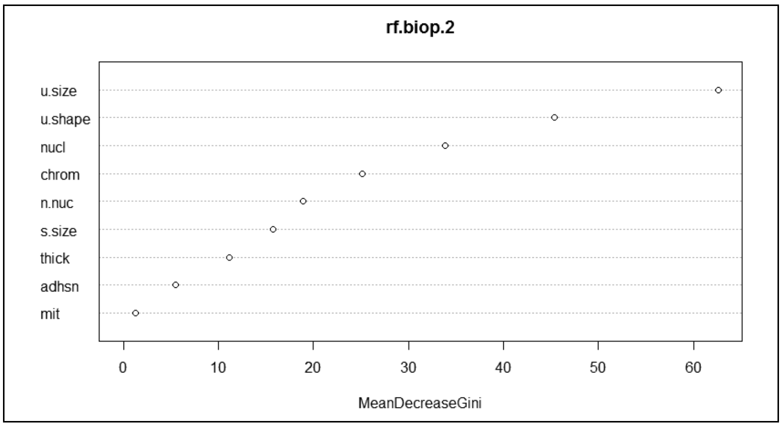
只需19棵树就可以使模型正确率达到最优。实验一下，看看模型的表现：

```
> set.seed(123)
> rf.biop.2 <- randomForest(class~ ., data = biop.train, ntree =
 19)
> print(rf.biop.2)
Call:
 randomForest(formula = class ~ ., data = biop.train, ntree = 19)
        Type of random forest: classification
           Number of trees: 19
No. of variables tried at each split: 3
    OOB estimate of error rate: 2.95%
Confusion matrix:
     benign malignant class.error
benign    294      8 0.02649007
malignant   6    166 0.03488372
> rf.biop.test <- predict(rf.biop.2, newdata = biop.test, type =
   "response")
> table(rf.biop.test, biop.test$class)
rf.biop.test benign malignant
  benign    139       0
  malignant  3      67
> (139 + 67) / 209
[1] 0.9856459
```

怎么样？训练集上的误差率还不到3%，在测试集上甚至表现得更好。测试集的209个观测中，只有3个观测被误分类，而且没有一个是误诊为“恶性”的。回忆一下，我们使用逻辑斯蒂回归得到的正确率最好结果是97.6%，所以随机森林模型就是至今为止在乳腺癌数据上的最佳模型。进行下面的内容之前，先看看变量重要性统计图。

```
> varImpPlot(rf.biop.2)
```

上述命令输出如下。



6

上页图中，变量重要性是指每个变量对基尼指数平均减少量的贡献，此处的变量重要性与单个树分裂时有很大区别。回忆一下，单个树是在细胞大小均匀度开始分裂的（与随机森林一致），然后是nuclei，接着是细胞密度。这揭示了随机森林技术具有非常大的潜力，不但可以提高模型预测能力，还可以改善特征选择的结果。

现在，开始面对皮玛印第安人糖尿病模型的艰巨挑战。我们先做好数据准备，如下所示：

```
  > data(Pima.tr)
> data(Pima.te)
> pima <- rbind(Pima.tr, Pima.te)
> set.seed(502)
> ind <- sample(2, nrow(pima), replace = TRUE, prob = c(0.7, 0.3))
> pima.train <- pima[ind == 1, ]
> pima.test <- pima[ind == 2, ]
```

做完数据准备工作，下一步建立模型。如下所示：

```
  > set.seed(321)
> rf.pima = randomForest(type~., data=pima.train)
> rf.pima
Call:
 randomForest(formula = type ~ ., data = pima.train)
        Type of random forest: classification
           Number of trees: 500
No. of variables tried at each split: 2
    OOB estimate of error rate: 20%
Confusion matrix:
   No Yes class.error
No 233 29  0.1106870
Yes 48 75  0.3902439
```

我们在训练数据集上得到了20%的误分类误差率，并不比以前的结果更好。下面对树的数目进行优化，看看能否显著改善模型结果：

```
  > which.min(rf.pima$err.rate[, 1])
[1] 80
> set.seed(321)
> rf.pima.2 = randomForest(type~., data=pima.train, ntree=80)
> print(rf.pima.2)
Call:
 randomForest(formula = type ~ ., data = pima.train, ntree = 80)
        Type of random forest: classification
           Number of trees: 80
No. of variables tried at each split: 2
    OOB estimate of error rate: 19.48%
Confusion matrix:
   No Yes class.error
No 230 32  0.1221374
Yes 43 80  0.3495935
```

当随机森林中有80棵树时，OOB误差有些许改善。随机森林能够在测试集上证明自己吗？我

们来看一下。如下所示：

```
  > rf.pima.test <- predict(rf.pima.2, newdata= pima.test,
   type = "response")
> table(rf.pima.test, pima.test$type)
rf.pima.test No Yes
     No 75 21
     Yes 18 33
> (75+33)/147
[1] 0.7346939
```

我们在测试集上只得到了73%的正确率，低于使用SVM的结果。

虽然随机森林在糖尿病数据集上令人失望，但它已经证明了，自己在乳腺癌诊断方面是迄今为止最好的分类器。最后，我们实验一下梯度提升方法。

5. 极限梯度提升——分类

我们提到过，在这一节要使用已经加载的xgboost包。因为这种方法的效果如雷贯耳，所以我们直接用它解决最有挑战性的皮玛印第安人糖尿病问题。

正如在提升算法简介中所说，我们要调整一些参数。

- nrounds：最大迭代次数（最终模型中树的数量）。
- colsample_bytree：建立树时随机抽取的特征数量，用一个比率表示，默认值为1（使用100%的特征）。
- min_child_weight：对树进行提升时使用的最小权重，默认为1。
- eta：学习率，每棵树在最终解中的贡献，默认为0.3。
- gamma：在树中新增一个叶子分区时所需的最小减损。
- subsample：子样本数据占整个观测的比例，默认值为1（100%）。
- max_depth：单个树的最大深度。

> 使用expand.grid()函数可以建立实验网格，以运行caret包的训练过程。对于前面列出的参数，如果没有设定具体值，那么即使有默认值，运行函数时也会收到出错信息。下面的参数取值是基于我以前的一些训练迭代而设定的。建议各位亲自实验参数调整过程。

使用以下命令建立网格：

```
> grid = expand.grid(
 nrounds = c(75, 100),
 colsample_bytree = 1,
 min_child_weight = 1,
 eta = c(0.01, 0.1, 0.3), #0.3 is default,
 gamma = c(0.5, 0.25),
 subsample = 0.5,
```

```
max_depth = c(2, 3)
)
```

以上命令会建立一个具有24个模型的网格，caret包会运行这些模型，以确定最好的调优参数。此处必须提示各位，对于我们现在所用的这种规模的数据集，代码运行过程只需几秒钟，但对于一些大数据集，这个运行过程可能需要几小时。所以，在时间非常宝贵的情况下，你必须应用自己的判断力，并使用小数据样本来找出合适的调优参数，否则硬盘空间可能不足。

使用car包的`train()`函数之前，我要创建一个名为`cntrl`的对象，来设定`trainControl`的参数。这个对象会保存我们要使用的方法，以训练调优参数。我们使用5折交叉验证，如下所示：

```
> cntrl = trainControl(
 method = "cv",
 number = 5,
 verboseIter = TRUE,
 returnData = FALSE,
 returnResamp = "final"
 )
```

要使用`train.xgb()`函数，只需和训练其他模型一样，设定好所需参数即可：训练数据集、标号、训练控制对象和实验网格。记得要设定随机数种子：

```
> set.seed(1)
> train.xgb = train(
 x = pima.train[, 1:7],
 y = ,pima.train[, 8],
 trControl = cntrl,
 tuneGrid = grid,
 method = "xgbTree"
 )
```

因为在`trControl`中设定了`verboseIter`为`TURE`，所以可以看到每折交叉验证中的每次训练迭代。

调用`train.xgb`这个对象可以得到最优的参数，以及每种参数设置的结果，如下所示（有所简省）：

```
> train.xgb
eXtreme Gradient Boosting
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 308, 308, 309, 308, 307
Resampling results across tuning parameters:
eta max_depth gamma nrounds Accuracy Kappa
0.01 2     0.25  75    0.7924286 0.4857249
0.01 2     0.25  100    0.7898321 0.4837457
0.01 2     0.50  75    0.7976243 0.5005362
...............................................
0.30 3     0.50  75    0.7870664 0.4949317
0.30 3     0.50  100    0.7481703 0.3936924
```

```
Tuning parameter 'colsample_bytree' was held constant at a
value of 1
Tuning parameter 'min_child_weight' was held constant at a
value of 1
Tuning parameter 'subsample' was held constant at a value of 0.5
Accuracy was used to select the optimal model using the largest
 value.
The final values used for the model were nrounds = 75, max_depth =
 2,
eta = 0.1, gamma = 0.5, colsample_bytree = 1, min_child_weight = 1
and subsample = 0.5.
```

由此可以得到最优的参数组合来建立模型。模型在训练数据上的正确率是81%，Kappa值是0.55。下面要做的事情有点复杂，但我认为这才是最佳实践。首先创建一个参数列表，供Xgboost包的训练函数`xgb.train()`使用。然后将数据框转换为一个输入特征矩阵，以及一个带标号的数值型结果列表（其中的值是0和1）。接着，将特征矩阵和标号列表组合成符合要求的输入，即一个`xgb.Dmatrix`对象。代码如下：


```
> param <- list( objective = "binary:logistic",
 booster = "gbtree",
 eval_metric = "error",
 eta = 0.1,
 max_depth = 2,
 subsample = 0.5,
 colsample_bytree = 1,
 gamma = 0.5
 )
> x <- as.matrix(pima.train[, 1:7])
> y <- ifelse(pima.train$type == "Yes", 1, 0)
> train.mat <- xgb.DMatrix(data = x, label = y)
```

这些准备工作完成之后，即可创建模型：

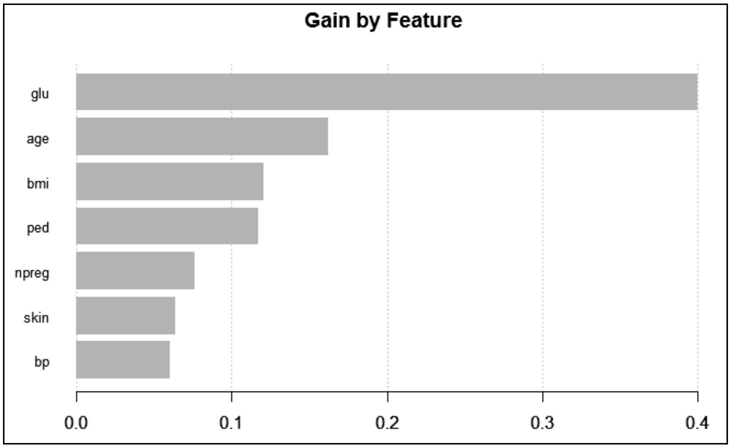
```
> set.seed(1)
> xgb.fit <- xgb.train(params = param, data = train.mat, nrounds =
 75)
```

在测试集上查看模型效果之前，先检查变量重要性，并绘制统计图。你可以检查3个项目：gain、cover和frequecy。gain是这个特征对其所在分支的正确率做出的改善，cover是与这个特征相关的全体观测的相对数量，frequency是这个特征在所有树中出现的次数百分比。以下代码会生成我们需要的输出：

```
> impMatrix <- xgb.importance(feature_names = dimnames(x)[[2]],
 model = xgb.fit)
> impMatrix
  Feature      Gain     Cover Frequency
1:    glu 0.40000548 0.31701688 0.24509804
2:    age 0.16177609 0.15685050 0.17156863
3:    bmi 0.12074049 0.14691325 0.14705882
4:    ped 0.11717238 0.15400331 0.16666667
```

```
5:  npreg 0.07642333 0.05920868 0.06862745
6:  skin 0.06389969 0.08682105 0.10294118
7:   bp 0.05998254 0.07918634 0.09803922
> xgb.plot.importance(impMatrix, main = "Gain by Feature")
```

上述命令输出如下。



与其他方法相比，这个特征重要性图有什么不同吗？

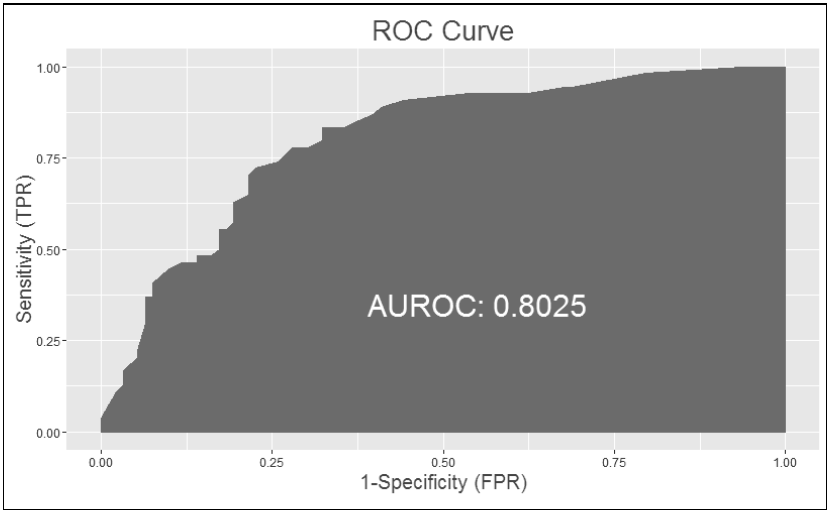
下面看看这个模型在测试集上的表现。与训练集一样，测试集数据也要转换为矩阵。再次使用InformationValue包中的工具来帮助我们完成这个任务。下面的代码先加载程序库，再生成预测结果来分析模型性能：

```
> library(InformationValue)
> pred <- predict(xgb.fit, x)
> optimalCutoff(y, pred)
[1] 0.3899574
> pima.testMat <- as.matrix(pima.test[, 1:7])
> xgb.pima.test <- predict(xgb.fit, pima.testMat)
> y.test <- ifelse(pima.test$type == "Yes", 1, 0)
> confusionMatrix(y.test, xgb.pima.test, threshold = 0.39)
  0 1
0 72 16
1 20 39
> 1 - misClassError(y.test, xgb.pima.test, threshold = 0.39)
[1] 0.7551
```

你注意到我在代码中使用`optimalCutoff()`函数做的事情了吗？这个Information包中的函数可以找出使误差最小化的最优概率阈值。顺便说一下，模型误差大概是25%，还是没有超过SVM模型。再悄悄告诉你，可以看看ROC曲线，AUC超过了0.8。以下代码可以生成ROC曲线：

```
> plotROC(y.test, xgb.pima.test)
```

代码输出如下页图。



6.2.2 模型选择

回忆一下我们在本章的基本目标——使用基于树的学习方法提高前几章中得到的模型的预测能力。我们有什么收获？首先，在具有定量响应变量的前列腺癌数据集上，第4章已经得到了一个线性模型，但我们的努力没有改善这个模型。其次，随机森林在威斯康星乳腺癌数据集上的表现超过了第3章中的逻辑斯蒂回归模型。最后，我必须失望地宣布，在皮玛印第安人糖尿病数据集上，使用提升树不能得到比SVM模型更好的结果。

综上，使我感到欣慰的是，对于前列腺癌数据集和乳腺癌数据集，我们得到了比较好的模型。第7章将介绍神经网络和深度学习的概念，我们会再次尝试改善糖尿病模型。开始下一章之前，我想介绍一种使用随机森林技术的强大的特征消除方法。

6.2.3 使用随机森林进行特征选择

到现在为止，我们已经介绍了几种特征选择技术，比如正则化、最优子集以及递归特征消除。下面要介绍一种对于分类问题非常有效的特征选择方法，这种方法是通过随机森林实现的，需要使用Boruta包。有一篇论文详细介绍了这种方法如何选择所有相关特征：

Kursa M., Rudnicki W. (2010), Feature Selection with the Boruta Package, *Journal of Statistical Software*, 36(11), 1 - 13

在这一节，我会简单介绍这种算法，然后将其应用于一个特征非常多的数据集。我发现这种算法非常有效，但有人警告我，说这种算法非常消耗计算能力，可能达不到预想的效果。但它确实可以有效消除不重要的特征，使我们可以集中精力构建一个更简洁、更有效，也更有意义的模

型，还是值得花时间学习的。

在更高的层次上，算法会复制所有输入特征，并对特征中的观测顺序进行重新组合，以去除相关性，从而创建**影子特征**。然后使用所有输入特征建立一个随机森林模型，并计算每个特征（包括影子特征）的正确率损失均值的Z分数。如果某个特征的Z分数显著高于影子特征的Z分数，那么这个特征就被认为是**重要**的；反之，这个特征就被认为是**不重要**的。然后，去除掉影子特征和那些已经确认了重要性的特征，重复上面的过程，直到所有特征都被赋予一个表示重要性的值。你可以设定随机森林迭代的最大次数。算法结束之后，每个初始特征都会被标记为**确认**、**待定**或**拒绝**。对于待定的特征，你必须自己确定是否要包括在下一次建模中。根据具体情况，你可以有以下几种选择：

- 改变随机数种子，重复运行算法多次（*k*次），然后只选择那些在*k*次运行中都标记为“确认”的属性；
- 将你的训练数据分为*k*折，在每折数据上分别进行算法迭代，然后选择那些在所有*k*折数据上都标记为“确认”的属性。

请注意，所有这些工作都可以用几行代码完成。下面查看代码，并将其应用于Sonar数据集，这个数据集来自mlbench包。数据集中有208个观测，60个输入特征，以及1个用于分类的标号向量。标号是个因子，如果sonar对象是岩石，标号就是R；如果sonar对象是矿藏，标号则是M。首先加载数据，并快速进行数据探索：

```
> data(Sonar, package="mlbench")
> dim(Sonar)
[1] 208 61
> table(Sonar$Class)
  M  R
111 97
```

要运行这个算法，你只须加载Boruat包，然后在`boruta()`函数中创建一条模型公式。请记住，标号必须是因子类型，否则算法不会正常执行。如果想跟踪算法的进程，可以设定`doTrace = 1`。还有，不要忘了设定随机数种子：

```
> library(Boruta)
> set.seed(1)
> feature.selection <- Boruta(Class ~ ., data = Sonar, doTrace = 1)
```

正如我们前面提到过的，这个算法需要大量的计算能力。在我的老式笔记本电脑上，需要以下这么长的时间：

```
> feature.selection$timeTaken
Time difference of 25.78468 secs
```

从一个简单的表格可以得出最终重要决策的计数。可以看出，我们完全能够去除大约一半特征：

```
> table(feature.selection$finalDecision)
Tentative Confirmed Rejected
    12    31    17
```

根据以上结果，可以很容易地使用我们选择的特征创建一个新的数据框。从`getSelectedAttributes()`函数开始，这个函数可以找出特征名称。在下面的例子中，只选择那些“确认”的特征。如果想包括“确认”和“待定”的特征，只需在函数中指定`withTentative=TRUE`：

```
  > fNames <- getSelectedAttributes(feature.selection) # withTentative =
TRUE
  > fNames
[1] "V1" "V4" "V5" "V9" "V10" "V11" "V12" "V13" "V15" "V16"
[11] "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V27" "V28" "V31"
[21] "V35" "V36" "V37" "V44" "V45" "V46" "V47" "V48" "V49" "V51"
[31] "V52"
```

使用这些特征名称，可以创建一个Sonar数据集的子集：

```
> Sonar.features <- Sonar[, fNames]
> dim(Sonar.features)
[1] 208 31
```


一切搞定！数据框Sonar.features包含了boruta算法选择的所有“确认”特征。现在可以使用它进行更深入、更有意义的数据探索。通过几行代码和一些对算法运行的耐心等待，即可显著提高建模工作和知识生成的水平。

6.3 小结

你在本章既体会了基于树的学习方法在处理分类和回归问题方面的威力，也看到了它们的局限性。单个树模型虽然易于构建和解释，但对于我们试图解决的多数问题不具备必需的预测能力。要想提高模型的预测能力，建议使用随机森林和梯度提升树。通过随机森林方法可以建立几十棵甚至几百棵树，这些树的结果会聚集成一个综合的预测。随机森林中的每棵树都是使用数据抽样建立的，抽样的方法称为自助法，预测变量也同样进行抽样。对于梯度提升方法，先创建一棵初始的、相对小规模的树，之后，基于残差或误分类会生成一系列树。这种技术的预期结果是建立一系列树，后面的树可以对前面的树的缺点加以改善，最终降低偏差和方差。我们还知道，在R中，可以使用随机森林作为特征选择的方法。

尽管这些方法确实非常强大，但在机器学习世界中它们不是万能的。不同的数据集要求分析者根据实际情况判断应该使用什么分析技术。对于分析者来说，技术的选择和调优参数的选择同等重要，有些预测模型是好的，但有些预测模型是伟大的，这种不断调整和细化的过程决定了二者之间的所有区别。

下一章的主要任务是，使用R建立神经网络和深度学习模型。

第 7 章

神经网络与深度学习

“在大数据这个勇敢新世界中，忘了人工智能吧，我们应该小心不要成为‘人工智障’。”

——汤姆·查特菲尔德

大概是2012年年中，我参加了一个关于某个分析结果的小组讨论。人们围坐在桌旁，有个家伙突然喊了起来：“这不就是什么神经网络吗？不是吗？”语气中有一丝丝恼怒，还有一点点惊恐。我知道他以前对神经网络非常抵触，还有着根深蒂固的焦虑，于是，为了平息他的忧虑，我发表了一通尖刻的评论，说神经网络已经像恐龙一样走在灭绝的道路上了。居然没有人表示反对！几个月之后，我列席了一个内部会议，会议的中心内容令我目瞪口呆，最让我惊奇的就是神经网络和不可思议的深度学习。吴恩达、杰弗里·辛顿、鲁斯兰·萨拉克霍特迪诺夫和约书亚·本吉奥这些机器学习的先行者们已经使神经网络起死回生，重新焕发了生机与活力。

很多媒体都在热炒神经网络和深度学习，把它们同各种高科技公司联系在一起，比如Facebook、谷歌或Netflix，这些公司在神经网络技术上投资了几千万甚至几亿美元。在语音识别、图像识别、机械制造和自动化领域，神经网络技术也取得了预期的成果。如果自动驾驶汽车不在公路上横行霸道并且彼此撞做一团的话，我也很愿意把它放在我们要讨论的主题中。

我们将在本章讨论神经网络的使用方法和优缺点，以便你能够真正理解和掌握这种技术。我们会详细介绍如何使用神经网络完成一个实际的商业应用，并在一个基于云的应用上面使用深度学习方法。

7.1　神经网络介绍

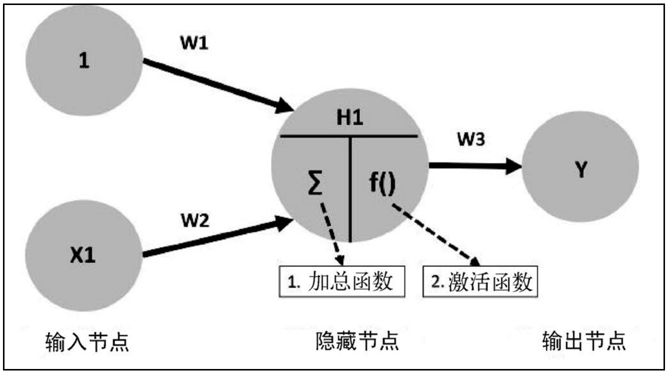
“神经网络”的概念相当宽泛，它包括了很多相关的方法。在本书中，我们主要关注使用**反向传播**方法进行训练的**前馈**神经网络。这种机器学习方法与生物学意义上的大脑运作方式“和而不同”，但我不准备浪费时间讨论。我们只需从神经网络的通用定义开始，你最好先看看维基百科上的条目。

在机器学习与认知科学领域，**人工神经网络**是一系列统计学习模型，它的灵感来自于生物神经网络（动物中央神经系统，特别是大脑），用来估计或近似那些依赖大量输入且通常未知的功能（https://en.wikipedia.org/wiki/Artificial_neural_network）。

开发神经网络模型的动机（或者说神经网络模型的优点）在于，可以对输入变量（特征）和响应变量之间的高度复杂关系进行建模，特别是关系呈现高度非线性时。神经网络模型的构建和评价不需要基本假设，对于定量和定性响应变量都适用。以上就是神经网络中的“阴”，下面看看其中的“阳”。对神经网络的一种常见的批评就是，它的结果是一个黑盒。换言之，没有一个带有系数的等式可供检验并分享给业务伙伴。实际上，结果几乎是无法解释的。另外一种批评意见的主要内容是，当初始的随机输入发生变化时，我们不清楚结果会发生什么变化。还有，神经网络的训练过程需要昂贵的时间和计算成本。

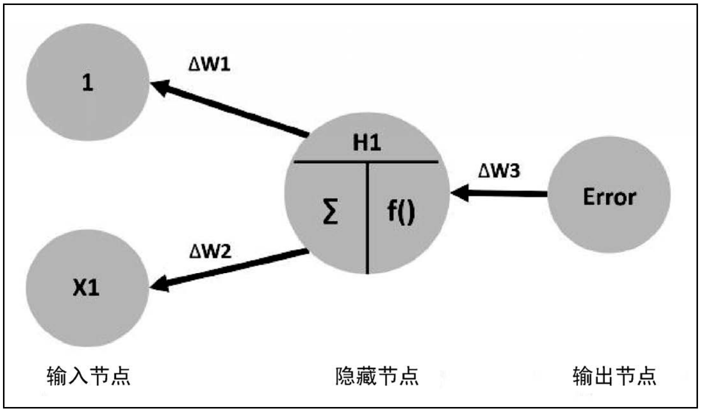
神经网络背后的数学理论非常艰深，尽管如此，我们至少应该理解它是如何运转的，这一点非常重要。我们从一个极其简单的神经网络图解开始，这是直观理解神经网络的一个好方式。

在这个简单的网络中，输入（又称协变量）由两个节点（又称神经元）组成。标有1的神经元代表一个常数，更确切地说，是截距。X_1代表一个定量变量，**W**代表权重，会和输入节点的值相乘。**输入节点**的值在与W相乘后传递到**隐藏节点**。你可以有多个隐藏节点，但是工作原理和一个隐藏节点没有什么不同。在隐藏节点H_1中，所有加权后的输入值被加总。因为截距为1，所以这个输入节点的值就是权重W_1。然后就是见证奇迹的时刻：加总后的值通过**激活函数**进行转换，将输入信号转换为输出信号。在这个简单例子中，H_1是唯一的隐藏节点，它的值被乘以W_3，成为响应变量Y的估计值。这就是算法的前馈部分。



等等，还没完！要完成这个循环，或者用神经网络中的说法，完成一次完整的“训练”，还要进行反向传播过程，基于学习到的知识来训练模型。为了初始化反向传播过程，需要基于损失函数确定误差，损失函数可以是**误差平方总和**，也可以是**交叉熵**，或者其他形式。因为权重$\mathbf{W_1}$和$\mathbf{W_2}$最初被设定为[−1, 1]之间的随机数，所以初始的误差可能会很大。反向传播时，要改变权重值以使损失函数中的误差最小。下页的图描述了算法的反向传播部分。

7



这样就完成了一次完整的训练。这个过程不断继续，使用梯度下降方法（第5章）减小误差，直到算法收敛于误差最小值或者达到预先设定的训练次数。如果假设本例中的激活函数是简单线性的，那么结果为$Y = W_3(W_1(1) + W_2(X_1))$。

如果你增加了大量的输入神经元，或者在隐藏节点中增加多个神经元，甚至增加多个隐藏节点，神经网络会变得非常复杂。请一定注意，一个神经元的输出会连接到所有后继的神经元，并将权重分配给所有连接，这样会大大增加模型的复杂性。增加隐藏节点和提高隐藏节点中神经元的数量不会像我们希望的那样改善模型的性能。于是，深度学习得以发展，它可以部分放松所有神经元都要连接的要求。

有很多种激活函数可供使用，其中包括一个简单线性函数，还有用于分类问题的sigmoid函数，它是逻辑斯蒂函数的一种特殊形式（第3章）。当输出变量是基于某种阈值的二值变量（0或1）时，可以使用阈值函数。其他常用的激活函数还有Rectifier、Maxout以及**双曲正切函数**（tanh）。

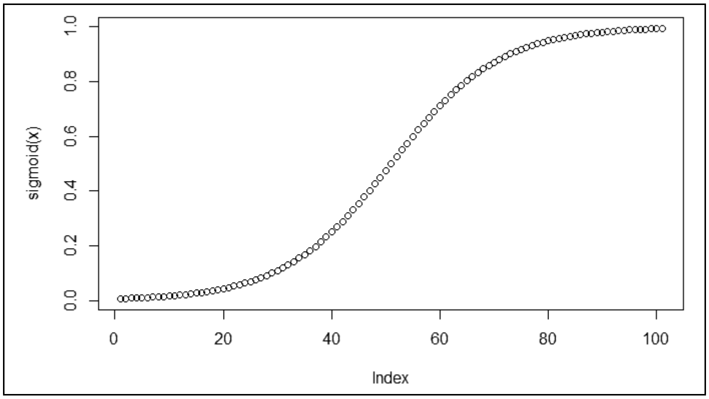
可以用R画出sigmoid函数。先创建一个R函数，计算sigmoid函数值：

```
   > sigmoid = function(x) {
1 / ( 1 + exp(-x) )
}
```

然后，在某个取值区间（比如–5～5）画出函数就非常简单了：

```
> x <- seq(-5, 5, .1)
> plot(sigmoid(x))
```

上述命令输出如下页图。



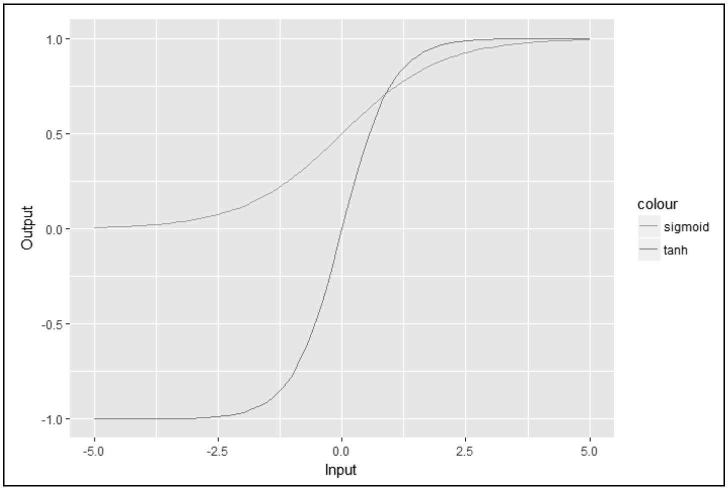
`tanh()`函数（双曲正切函数）是sigmoid函数的一种变体，它的输出值为−1 ~ 1。tanh函数和sigmoid函数的关系如下所示：

$$\tanh(x) = 2 * \text{sigmoid}(2x) - 1$$

绘制并比较tanh函数和sigmoid函数。仍然使用ggplot：

```
> library(ggplot2)
> s <- sigmoid(x)
> t <- tanh(x)
> z <- data.frame(cbind(x, s, t))
> ggplot(z, aes(x)) +
  geom_line(aes(y = s, color = "sigmoid")) +
  geom_line(aes(y = t, color = "tanh"))
```

上述命令输出如下。



既然有了sigmoid函数，为什么还要使用tanh函数呢？tanh函数在神经网络中很流行吗？对于

这些问题，真可谓众说纷纭。简言之，假设你对数据做了缩放处理，使数据的均值为0、方差为1，那么tanh函数就可以提供一组均值接近于0（以0为中心）的权重。这样的权重有助于避免偏差，并可以提高收敛速度。想象一下，如果使用sigmoid函数作为激活函数，那么就意味着从输出神经元到输入神经元都是正的权重。在后向传播过程中，这样就会导致各层之间的权重或者都是正的，或者都是负的，从而引发性能方面的问题。还有，因为sigmoid函数在两个尾部（0和1）的梯度接近于0，所以在后向传播过程中，几乎没有信号在不同层次的神经元之间流动。对于这个问题的详细讨论可以参见LeCun（1998）。请注意，“tanh函数效果更好”也并非定论。

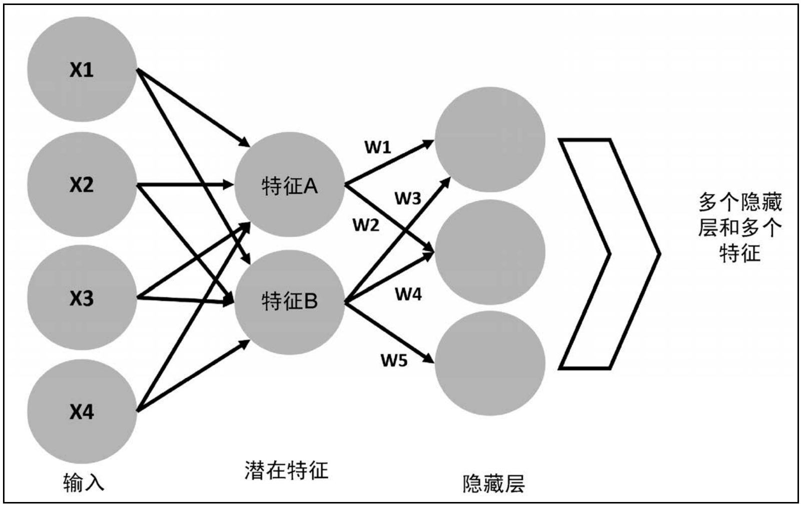
这一切听上去真令人心驰神往，但ANN表现得并不像传说中那么好，特别在你试图使用具有多个隐藏层和神经元的深度网络时，尤为如此。随着杰弗里·辛顿和鲁斯兰·萨拉克霍特迪诺夫2006年那篇论文的发表，神经网络又披着一层新的外衣逐渐成为热门。但让我说的话，这不过是对神经网络的重新包装——深度学习。

7.2　深度学习简介

那么，到底什么是深度学习？为什么它能吸引如此多的注意？抢占如此多头条？我们还是先看看维基百科上的通用定义：深度学习是机器学习的一个分支，它基于一整套算法，试图对数据中的高层次抽象进行建模，建模过程需要使用带有复杂结构或其他结构的由多重非线性转换组成的模型体系结构。这个定义看上去出自某位律师之手。深度学习的基础就是神经网络，它的特点其实就是使用机器学习技术（一般是无监督学习）在输入变量的基础之上构建新的特征。在后面的章节中，我们要对一些无监督学习技术进行深入研究。现在，你可以认为它们就是在没有响应变量的情况下寻找数据中的结构或模式。**元素周期表**就是一个经典的例子，它在没有指定响应变量时找出数据结构。你展开元素周期表就会看到，它是按照原子结构排列，金属在一侧，非金属在另一侧，基于潜在分类（结构）建立。这种对潜在结构（层次）的识别就是深度学习与那些千篇一律的ANN的区别所在。相比于那些只有原始输入的问题，深度学习应该更适用于那种可以通过算法对输出结果进行更好表示的问题。换句话说，模型在仅有原始像素作为唯一输入的情况下，能够学会对照片进行分类吗？当你有少量标记过的响应变量和海量未标记的输入数据时，深度学习非常有用。你可以通过无监督学习方法训练深度学习模型，然后通过监督式方法将其应用在标记过的数据上，这样向前/向后不断迭代。

要用数学解释这种潜在结构的识别是非常有难度的，但我们在第4章介绍过的正则化概念正是一个例子。在深度学习中，你可以使用正则化方法对权重进行惩罚，比如L1（惩罚非0权重）、L2（惩罚过大权重）和丢弃（随机忽略某种输入，将其权重归零）。标准神经网络中不使用这些正则化技术。

另外一种识别潜在结构的方式是降低数据的维度，比如自动编码器。在这种神经网络中，输入被转换为一组降低了维度的权重。请注意，下页图中的**特征A**和一个隐藏节点之间没有连接。



这种方法可以递归使用，学习可以在多个隐藏层之间进行。你在这个例子中能发现的就是，网络在不断使用原有特征制造新特征，以至于它们互相堆叠在一起。深度学习会先在两个层之间按前后顺序学习权重，然后使用反向传播方法对这些权重进行微调。其他的特征选择方法包括**受限波尔兹曼机和稀疏编码模型**。

深度学习技术的细节已经超出本书范围，想要深入学习的话，有很多资源可以使用。你可以从下面两个网站开始：

- http://www.cs.toronto.edu/~hinton/
- http://deeplearning.net/

深度学习在很多分类问题上表现得非常好，还赢得了一两次Kaggle竞赛。和神经网络一样，它也受到一些问题的困扰，最主要的就是黑盒问题。你可以试着向不明情况的人解释一下神经网络中都发生了什么。尽管如此，对于那种“不管黑猫白猫，捉住老鼠就是好猫”的问题，深度学习是非常适合的。归根结底，我们真的想知道自动驾驶汽车为什么可以避开行人吗？或者说，我们更关注的难道不是自动驾驶汽车是否撞到了行人吗？多说一句，Python社区在深度学习的使用和软件开发方面领先R社区一步。我们在后面的实战练习中会看到，这个距离正在逐渐缩小，但还没有完全消除。

不管怎么说，深度学习都是一种激动人心的技术。应该注意的是，要想发挥这项技术的全部威力，你不但需要强大的计算能力，还需要花费大量时间，通过微调超参数去训练最优模型。以下是一些注意事项：

- 激活函数
- 隐藏层的规模

- 降低维度的方法（受限波尔兹曼机还是自动编码器）
- 完整训练的次数
- 梯度下降学习率
- 损失函数
- 正则化

深度学习资源与高级方法

TensorFlow™提供了一个可交互的窗口部件，地址是http://playground.tensorflow.org/。这个可视化工具非常有趣，你可以用它进行学习和解释。对于分类问题或回归问题，你可以使用这个工具探索（这个站点称为tinker）不同的参数组合，看看它们对响应变量的影响。我愿意（实际上已经）花费数小时使用这个工具进行探索。

TIP　交给你一项非常有意思的任务：自己设计一个实验，看看不同的参数组合如何影响你的预测。

现在，TensorFlow™和MXNet似乎是成长最快的两种深度学习工具。我还是选择使用H2O包来工作，这个程序包会在后面介绍。但是，学习和理解最新技术是非常重要的。你可以通过R使用TensorFlow™，但需要先安装Python。下面的这个系列教程会指导你如何完成这个任务：

https://rstudio.github.io/tensorflow/

MXNet并不要求安装Python，它的安装和使用也比较容易。MXNet还提供了很多已经训练好的模型，可以使你快速开始预测。下面的网址中有一些R的教程：

http://mxnet.io/

下面，我要花一些篇幅列举深度神经网络的几个变体，以及它们适用的学习任务。

卷积神经网络：前提假设是输入为图像，并使用一小部分数据或数据切片来创建特征，并将特征组合创建特征映射图。这些小数据切片可以作为网络在训练过程中学习的过滤器，或者更确切地说，是卷积核。CNN的激活函数是**线性整流函数**，它的简单形式是$f(x) = \max(0, x)$，其中x是神经元的输入。CNN在图像分类、目标检测甚至句子分类方面的效果非常好。

循环神经网络：创建RNN的目的是利用序贯信息。在传统神经网络中，输入与输出之间是互相独立的。而在RNN中，输出则依赖于上一层的计算结果，允许信息在层与层之间保持。所以，对于一个神经元的输出（y），并不仅是根据它的输入计算出来的，还要包括所有上层的输出（$t-1$、$t-n\cdots$）。RNN对于笔迹检测和语音检测特别有效。

长短期记忆网络：LSTM是RNN的一种特殊形式。RNN有一个问题，它在具有长期信号的数据上表现不佳。于是，数据科学家们创建了LSTM来捕获数据中的复杂模式。在训练过程中，RNN

组合信息时，对来自以前步骤的信息以同样的方式进行处理，没有考虑不同步骤中的信息在价值上多少会有些不同。LSTM通过确定在训练过程中的每个步骤要记住哪些信息，以此克服RNN的这种局限性。这时的权重矩阵通过数据向量实现了倍增，在LSTM中称为“门”，它的作用就是过滤信息。LSTM中的神经元有两个输入和两个输出，它接受来自上一层神经元的输出和从前一个门传递过来的记忆向量，然后生成输出值和输出记忆向量，作为下一层的输入。LSTM的局限性在于，它需要完整清晰的训练数据以及非常高的计算能力。LSTM在语音识别问题上的效果非常好。

TIP　我建议你学习MXNet的教程，它有助于理解这些模型并为你所用。

好了，下面我们看一些实际的应用。

7.3　业务理解

1998年4月20日，一个平静祥和的夜晚，我作为修斯500D直升机上的一名见习飞行员，正在进行长途飞行，从明尼苏达州圣保罗市中心机场返回北达科他州大福克斯，回到熟悉又温馨的家乡。这是我通过考试并获得直升机仪表等级要求的最后一次飞行。飞行日志显示，此时我们距离2号航道指示标上的VOR导航系统还有35DME（Distance Measuring Equipment，测距装置），或者说是35海里。这说明，我们正位于明尼苏达州圣克劳德市南部（或东南部）的某个地方，在高于海平面4500英尺的高度（我记忆中是这样的）以120节的速度巡航。然后，事故发生了……砰！毫不夸张地说，就像一个巨雷在头上炸响，紧跟着，一团飓风似的气流拍在我的脸上。

这一切发生的时候，我的飞行教练正在问我一些基本问题，是关于进入明尼苏达州亚历山德里亚市的仪表进场着陆计划的。我们交换了飞机控制权，我弯下身去，查看放在膝板上的仪表进场过程图。就在我“啪”地一声打开红色闪光信号灯的那一刻，爆炸发生了。当时我正大头朝下，那声音，那接踵而至的气流，使我的脑海中一瞬间充斥着好几种想法：直升机要完蛋了，我就要死了；“挑战者”号航天飞机的爆炸就像高清电影一样出现在我的脑海里。1.359 秒之后，我们停止了尖叫，意识到自己面前的树脂挡风玻璃已经基本上没有了，但其他一切还算好。将飞机减速之后，我们仓促地检查了一下，发现直升机驾驶舱内满是鲜血、内脏和羽毛。真是不可思议，我们居然在明尼苏达中部的上空撞上了一只绿头野鸭，造成的后果就是挡风玻璃完全损坏。如果我没有弯腰查看膝板的话，就会被血肉糊一脸。我们向明尼阿波利斯指挥中心简单地告知了紧急情况，然后终止了飞行计划，像“孟菲斯美女”号一样，跌跌撞撞地飞到了亚历山德里亚市，等待来自北达科他大学的同胞的救援。

我为什么要啰嗦这些？好吧，我是想说我对美国国家航空航天局和宇航员有多么狂热的崇拜。在一个危机时刻，那一瞬间我认为这辈子已经完了，我的思维还漂移到航天飞机上去了。我这个年纪的多数男性都想和乔治·布雷特或韦恩·格雷茨基握手，我则希望同巴兹·奥尔德林握手。实际上，我也确实做到了（别忘了，他当时正任教于北达科他大学）。于是，当我发现MASS

包中有shuttle数据集时，认为必须在这本书中介绍它。顺便说一句，如果你有机会去肯尼迪航天中心参观“亚特兰蒂斯”号航天飞机，那千万不要错过。

在这个案例中，我们将开发一个神经网络模型，来回答“航天飞机是否应该使用自动着陆系统”这一问题。默认的决策是让机组人员控制飞船着陆，但在以下两种情况下需要自动着陆系统：一是当机组人员失去控制飞船的能力时；二是当飞船脱离轨道之后重新进入轨道，从而需要对抗重力时。数据是由计算机模拟产生的，不是从实际的飞行过程中得来的。实际上，自动着陆系统通过了一些非常严格的测试，绝大多数情况下，在飞船着陆过程中，由宇航员决定是否使用自动着陆系统。在以下的两个链接可以找到更多背景信息：

- http://www.sapceref.com/news/viewsr.html?pid=10518
- https://waynehale.wordpress.com/2011/03/11/breaking-through/

7.4　数据理解和数据准备

我们需要加载以下4个R包，数据保存在MASS包中：

```
> library(caret)
> library(MASS)
> library(neuralnet)
> library(vcd)
```

Neuralnet包用于构建模型，caret包用于数据准备，vcd包会帮助我们进行数据可视化。先加载数据，检查数据结构：

```
    > data(shuttle)
> str(shuttle)
'data.frame':256 obs. of  7 variables:
 $ stability: Factor w/ 2 levepicels "stab","xstab": 2 2 2 2 2 2 2
       2 2 2 ...
 $ error : Factor w/ 4 levels "LX","MM","SS",..: 1 1 1 1 1 1 1 1
       1 1 ...
 $ sign  : Factor w/ 2 levels "nn","pp": 2 2 2 2 2 2 1 1 1 1 ...
 $ wind  : Factor w/ 2 levels "head","tail": 1 1 1 2 2 2 1 1 1 2
       ...
 $ magn  : Factor w/ 4 levels "Light","Medium",..: 1 2 4 1 2 4 1
       2 4 1 ...
 $ vis   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1
       ...
 $ use   : Factor w/ 2 levels "auto","noauto": 1 1 1 1 1 1 1 1 1
1      ...
```

数据集包括256个观测和7个变量。请注意，所有变量都是分类变量，响应变量`use`有两个水平，`auto`和`noauto`。协变量如下所示。

- `stability`：能否稳定定位（`stab/xstab`）

- error：误差大小（MM/SS/LX）
- sign：误差的符号，正或负（pp/nn）
- wind：风向的符号（head/tail）
- magn：风力强度（Light/Medium/Strong/Out of Range）
- vis：能见度（yes/no）

我们将建立一些表格来探索数据，先从响应变量（结果）开始：

```
> table(shuttle$use)
auto noauto
 145    111
```

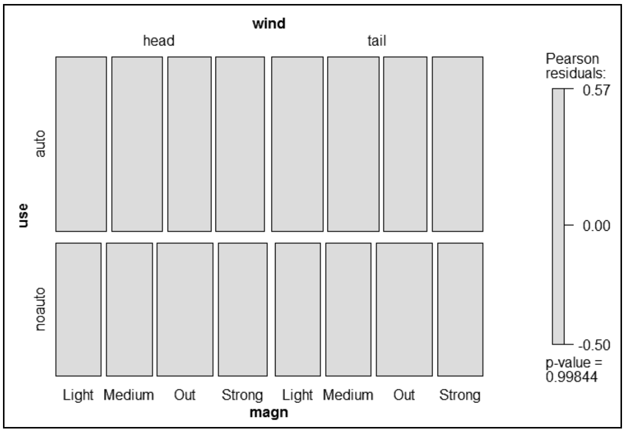
使用自动着陆的决策比例为57%。为分类数据建立表格有很多种方式，table()函数完美适用于两个变量之间的对比，但如果加入第三个变量，结果看上去就是一团糟。vcd包提供了很多制表和制图函数，其中一个是structable()，这个函数的语法公式是（列1 + 列2 ~ 列3），列3会在表格中以行的形式出现：

```
> table1 <- structable(wind + magn ~ use, shuttle)
> table1
       wind  head                      tail
       magn Light Medium Out Strong Light Medium Out Strong
use
auto           19     19  16     18    19     19  16     19
noauto         13     13  16     14    13     13  16     13
```

在表中，我们可以看出在逆风的情况下，如果风力为轻微（Light），那么自动着陆（auto）发生19次，非自动着陆（noauto）发生13次。vcd包提供了mosaic()函数，将structable()函数生成的表格绘制成统计图，同时提供了卡方检验的***p*值**。

```
> mosaic(table1, shading = T)
```

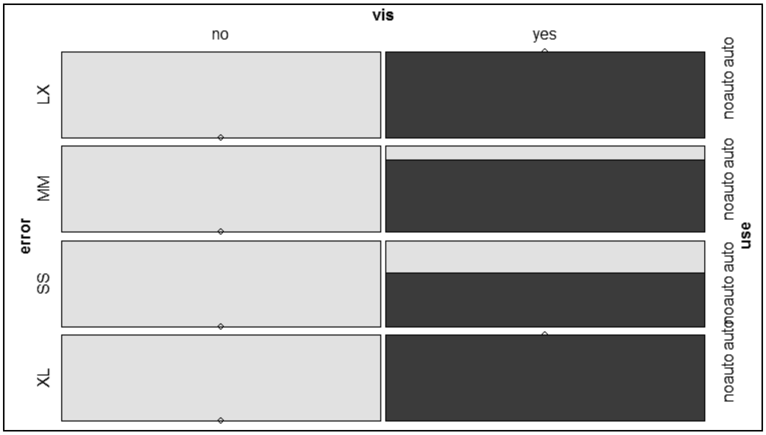
上述命令输出如下。



图中的方块可以表示表格中相应位置的数值比例，按递归的方式排列。还可以看到，*p*值是不显著的，所以特征与响应变量不相关。这说明，风力强度**magn**不能帮助我们预测是否使用自动着陆。如果要生成统计图，也可以不使用structable()建立的对象，因为mosaic()函数也可以接受公式：

```
> mosaic(use ~ error + vis, shuttle)
```

上述命令输出如下。



可以看到图中阴影发生了变化，反映出原假设被拒绝，变量之间具有相关性。统计图首先按照能见度（vis）进行划分，如果能见度为no，就要使用自动着陆。其次，按照**误差**（error）进行横向划分，如果误差为SS或MM，当vis为no时，就要使用自动着陆，其他情况则不使用自动着陆。这个图中不需要*p*值，因为深灰色的阴影已经指示得很明显了。

你还可以使用prop.table()函数查看比例表，将它作为table()函数的包装器即可：

```
    > table(shuttle$use, shuttle$stability)
         stab xstab
  auto     81    64
  noauto   47    64
> prop.table(table(shuttle$use, shuttle$stability))
              stab     xstab
  auto   0.3164062 0.2500000
  noauto 0.1835938 0.2500000
```

如果忘了*p*值，进行一次卡方检验也很容易：

```
    > chisq.test(shuttle$use, shuttle$stability)
Pearson's Chi-squared test with Yates' continuity
correction
data:  shuttle$use and shuttle$stability
X-squared = 4.0718, df = 1, p-value = 0.0436
```

神经网络的数据准备是非常重要的，因为所有协变量和响应变量都必须是数值型。在这个案例中，虽然所有变量都是分类变量，但caret包可以帮助我们快速建立虚拟变量作为输入特征：

```
    > dummies <- dummyVars(use ~ .,shuttle, fullRank = T)
> dummies
Dummy Variable Object
Formula: use ~ .
7 variables, 7 factors
Variables and levels will be separated by '.'
A full rank encoding is used
```

如果想把这个虚拟变量对象放到数据框中，需要使用虚拟变量预测现有数据（自身数据或其他数据），并通过as.data.frame转换。很显然，我们需要预测自身数据：

```
    > shuttle.2 = as.data.frame(predict(dummies, newdata=shuttle))

> names(shuttle.2)
 [1] "stability.xstab" "error.MM"        "error.SS"
 [4] "error.XL"        "sign.pp"         "wind.tail"
 [7] "magn.Medium"     "magn.Out"        "magn.Strong"
[10] "vis.yes"

> head(shuttle.2)
  stability.xstab error.MM error.SS error.XL sign.pp wind.tail
1               1        0        0        0       1         0
2               1        0        0        0       1         0
3               1        0        0        0       1         0
4               1        0        0        0       1         1
5               1        0        0        0       1         1
6               1        0        0        0       1         1
  magn.Medium magn.Out magn.Strong vis.yes
1           0        0           0       0
2           1        0           0       0
3           0        0           1       0
4           0        0           0       0
5           1        0           0       0
6           0        0           1       0
```

现在，我们得到具有10个变量的输入特征空间。对于stability，`0`表示`stab`，`1`表示`xstab`。error的基准是`LX`，用3个变量表示其他分类。

可以用`ifelse()`函数建立响应变量：

```
    > shuttle.2$use <- ifelse(shuttle$use == "auto", 1, 0)
> table(shuttle.2$use)
  0   1
111 145
```

caret包还可以生成训练数据集和测试数据集，做法是对每个观测赋一个索引标记，或者标记为训练数据，或者标记为测试数据，然后依照索引进行分类。我们按70/30的比例划分训练数据和测试数据，如下所示：

```
> set.seed(123)
> trainIndex <- createDataPartition(shuttle.2$use, p = .7, list =
    FALSE)
```

trainIndex中的值是行编号，在上面的代码中，是数据框shuttle.2中占总数70%的行编号。这样即可轻松建立训练集或测试集：

```
> shuttleTrain <- shuttle.2[trainIndex, ]
> shuttleTest <- shuttle.2[-trainIndex, ]
```

干得漂亮！下面开始建立神经网络模型。

7.5　模型构建与模型评价

前面提到过，我们要使用neuralnet包构建模型，其中建模函数使用的公式和其他方法一样，例如$y \sim x_1 + x_2 + x_3 + x_4$, data = df。以前，我们使用$y \sim$指定数据集中除响应变量之外的所有变量作为输入，但neuralnet中不允许这种写法。绕过这种限制的方式是，使用`as.formula()`函数。先建立一个保存变量名的对象，然后用这个对象作为输入，从而将变量名粘贴到公式右侧：

```
> n <- names(shuttleTrain)
> form <- as.formula(paste("use ~", paste(n[!n %in% "use"],
    collapse = " + ")))
> form
use ~ stability.xstab + error.MM + error.SS + error.XL + sign.pp +
    wind.tail
     + magn.Medium + magn.Out + magn.Strong + vis.yes
```

你应当记住这个函数的功能，因为使用起来实在是太方便了。在nerualnet包中，我们要使用的函数的名字就叫作`nerualnet()`。除了模型公式，还有4个关键参数需要说明。

- `hidden`：每层中隐藏神经元的数量，最多可以设置3个隐藏层，默认值为1。
- `act.fct`：激活函数，默认为逻辑斯蒂函数，也可以设置为tanh函数。
- `err.fct`：计算误差，默认为sse；因为我们处理的是二值结果变量，所以要设置成ce，使用交叉熵。
- `linear.output`：逻辑参数，控制是否忽略act.fct，默认值为`TRUE`；对于我们的数据来说，需要设置为`FALSE`。

还可以指定算法，默认算法是弹性反向传播。我们就使用这种算法，隐藏神经元也一样，使用默认值1：

```
> fit <- neuralnet(form, data = shuttleTrain, err.fct = "ce",
  linear.output = FALSE)
```

下页代码是整体结果：

```
  > fit$result.matrix
1
  error                           0.009928587504
  reached.threshold               0.009905188403
  steps                         660.000000000000
  Intercept.to.1layhid1          -4.392654985479
  stability.xstab.to.1layhid1     1.957595172393
  error.MM.to.1layhid1           -1.596634090134
  error.SS.to.1layhid1           -2.519372079568
  error.XL.to.1layhid1           -0.371734253789
  sign.pp.to.1layhid1            -0.863963659357
  wind.tail.to.1layhid1           0.102077456260
  magn.Medium.to.1layhid1        -0.018170137582
  magn.Out.to.1layhid1            1.886928834123
  magn.Strong.to.1layhid1         0.140129588700
  vis.yes.to.1layhid1             6.209014123244
  Intercept.to.use               30.721652703205
  1layhid.1.to.use              -65.084168998463
```

可以看到，误差非常低，仅为0.0099。steps的值是算法达到阈值所需的训练次数，也就是误差函数的偏导数的绝对值小于阈值（默认为0.1）时的训练次数。权重最高的神经元是vis.yes.to.1layhid1，权重值为6.21。

还可以看看所谓的广义权重。按照neuralnet包开发者的说法，广义权重被定义为第i个协变量对对数发生比的贡献：

> 广义权重表示每个协变量x_i的影响，可以类比为回归模型中的第i个回归系数。但是，广义权重依赖于所有其他协变量。（Gunther and Fritsch，2010）。

> **TIP** 可以调出并查看广义权重。我对输出结果进行了简化，只列出开始的4个变量和6个观测。请注意，如果你把每一行的数都相加，会得到一个相同的数值，这说明这种权重对每一种协变量组合都是相等的。请注意，由于权重随机初始化的原因，你的结果会有一些微小的差别。

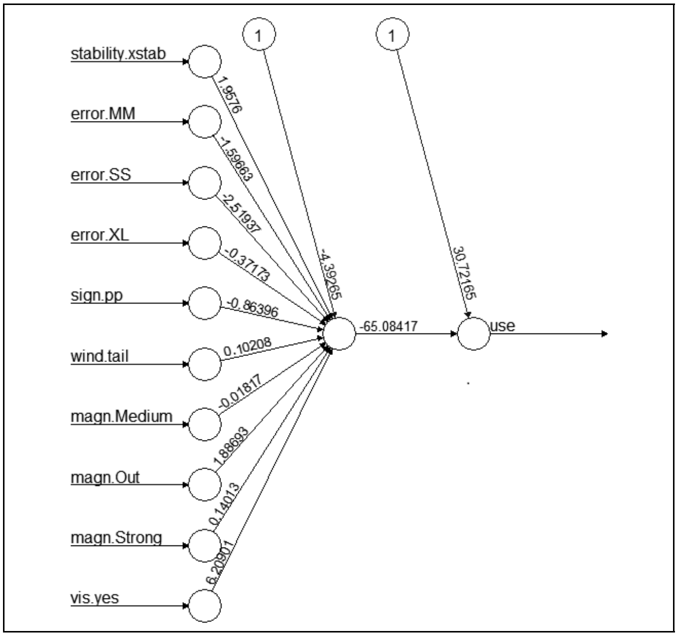
结果如下：

```
> head(fit$generalized.weights[[1]])
      [,1]             [,2]         [,3]          [,4]
 1 -4.374825405  3.568151106  5.630282059 0.8307501368
 2 -4.301565756  3.508399808  5.535998871 0.8168386187
 6 -5.466577583  4.458595039  7.035337605 1.0380665866
 9 -10.595727733 8.641980909 13.636415225 2.0120579565
10 -10.270199330 8.376476707 13.217468969 1.9502422861
11 -10.117466745 8.251906491 13.020906259 1.9212393878
```

要实现神经网络的可视化，只需使用`plot()`函数：

```
> plot(fit)
```

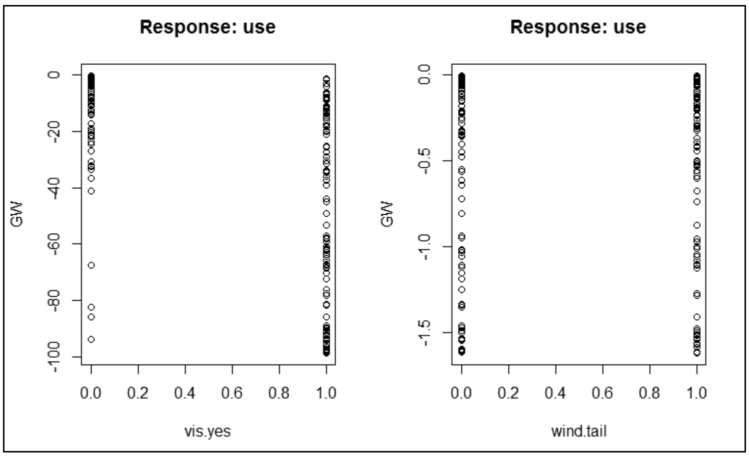
上述命令输出如下页图。



从这张统计图中，我们可以知道截距和每个变量的权重。你也可以在统计图中查看广义权重。下面看看vis.yes和wind.tail的广义权重图，wind.tail的连接权重在整体上处于较低的位置。请注意，vis.yes的广义权重非常不对称，而wind.tail的权重则分布得非常均匀，说明这个变量基本不具备预测能力：

```
    > par(mfrow = c(1, 2))
> gwplot(fit, selected.covariate = "vis.yes")
> gwplot(fit, selected.covariate = "wind.tail")
```

上述命令输出如下。



现在看看模型的表现如何，可以通过在compute()函数中指定fit模型和协变量来实现，语法与在测试集和训练集上进行预测是一样的。计算完成之后，通过$net.result会得到一个预测值列表：

```
    > resultsTrain <- compute(fit, shuttleTrain[, 1:10])
> predTrain <- resultsTrain$net.result
```

上面得到的结果是概率的形式，所以要将结果转换成0或1，然后生成混淆矩阵：

```
    > predTrain <- ifelse(predTrain >= 0.5, 1, 0)
> table(predTrain, shuttleTrain$use)
predTrain  0  1
        0 81  0
        1  0 99
```

天哪，神经网络模型的正确率达到了100%！屏住呼吸，看看它在测试集上的表现：

```
> resultsTest <- compute(fit, shuttleTest[,1:10])
> predTest <- resultsTest$net.result
> predTest <- ifelse(predTest >= 0.5, 1, 0)
> table(predTest, shuttleTest$use)
predTest   0   1
       0  29   0
       1   1  46
```

测试集中只有一个假阳性的误预测。如果你想找出这是哪个观测，可以使用which()函数：

7

```
    > which(predTest == 1 & shuttleTest$use == 0)
[1] 62
```

在测试集中，是第62行；在整个数据集中，是第203个观测。

能否建立一个正确率达到100%的神经网络模型？这个任务就交给你了！

7.6 深度学习示例

让我们结束对航天飞机的研究，使用H2O包完成一个深度学习的实战示例。我们要使用的数据是从加州大学欧文分校机器学习库中获取的，我对这份数据进行了一些处理，原始数据及其描述参见https://archive.ics.uci.edu/ml/datasets/Bank+Marketing。我所做的处理工作包括下载小数据集bank.csv，将数值型变量按比例缩放为均值为0、方差为1，为字符型变量或稀疏数值型变量创建虚拟变量，并删除方差基本为0的变量。这份数据可以在github上找到，地址为https://github.com/datameister66/data/，数据集名称是bank_DL.csv。我们在本节着重讨论如何将数据加载到H2O平台，并运行深度学习代码来建立一个分类器，预测客户是否会对一场营销活动做出反应。

7.6.1 H2O 背景介绍

H2O是一个开源的预测分析平台，它有很多预置算法，比如K最近邻、梯度提升机和深度学

习。你可以通过Hadoop、AWS、Spark、SQL、noSQL或自己的硬盘将数据上载到平台。H2O的一个巨大优点是，你可以在自己的本地计算机上使用平台上的大多数机器学习算法，这些算法是用R实现的。如果想知道更多细节，请访问站点http://h2o.ai/product/。

在R中安装H2O的过程有些与众不同。我在下面提供了安装最近更新的H2O版本（2017年2月25日）的代码。你可以使用这些代码重新安装最新版本，或者从http://h2o-release.s3.amazonaws.com/h2o/rel-lambert/5/docs-website/Ruser/Rinstall.html/下载。以下就是安装最新版本的代码：

```
# The following two commands remove any previously installed H2O
   packages for
  R.
 if ("package:h2o" %in% search()) { detach("package:h2o",
   unload=TRUE) }
 if ("h2o" %in% rownames(installed.packages())) {
   remove.packages("h2o") }

# Next, we download packages that H2O depends on.
if (! ("methods" %in% rownames(installed.packages()))) {
install.packages("methods") }
if (! ("statmod" %in% rownames(installed.packages()))) {
install.packages("statmod") }
if (! ("stats" %in% rownames(installed.packages()))) {
install.packages("stats") }
if (! ("graphics" %in% rownames(installed.packages()))) {
install.packages("graphics") }
if (! ("RCurl" %in% rownames(installed.packages()))) {
install.packages("RCurl") }
if (! ("jsonlite" %in% rownames(installed.packages()))) {
install.packages("jsonlite") }
if (! ("tools" %in% rownames(installed.packages()))) {
install.packages("tools") }
if (! ("utils" %in% rownames(installed.packages()))) {
install.packages("utils") }

# Now we download, install and initialize the H2O package for R.
install.packages("h2o", type="source", repos=(c("http://h2orelease.
s3.amazonaws.com/h2o/rel-tverberg/5/R")))
```

7.6.2　将数据上载到 H2O 平台

假设我们已经将bank_DL.csv文件保存到工作目录。请记住，`getwd()`函数会返回工作目录的路径。所以，下面可以加载程序库并创建一个对象，保存数据文件的路径：

```
> library(h2o)
> path <- "C:/.../bank_DL.csv"
```

现在可以连接到H2O平台，并在集群上启动一个实例。设置参数`nthreads = -1`，使实例可以使用集群上的所有CPU：

```
> localH2O = h2o.init(nthreads = -1)
```

利用H2O函数`h2o.uploadFile()`，可以将你的文件上载（导入）到H2O云平台。下面的函数也可以供你上载内容：

- ❑ `h2o.importFolder`
- ❑ `h2o.importURL`
- ❑ `h2o.importHDFS`

上传文件的过程非常简单，函数会给出一个表示百分比的进度条来跟踪上传状态：

```
  > bank <- h2o.uploadFile(path = path)
|===========================================================| 100%
```

上传的数据被保存在H2OFrame中，你可以通过`class()`函数检查，如下所示：

```
> class(bank)
[1] "H2OFrame"
```

在H2O平台中，很多R函数的输出和我们以前用过的函数不一样。比如，看一下表示数据结构的函数（我省略了一些输出）：

```
> str(bank)
Class 'H2OFrame' <environment: 0x0000000032d02e80>
  - attr(*, "op")= chr "Parse"
  - attr(*, "id")= chr "bank_DL_sid_95ad_2"
  - attr(*, "eval")= logi FALSE
  - attr(*, "nrow")= int 4521
  - attr(*, "ncol")= int 64
  - attr(*, "types")=List of 64
```

可以看到，有4521个观测（`nrow`）和64列（`ncol`）。顺便说一下，`head()`函数和`summary()`函数还和以前完全一样。划分数据集之前，看看响应变量的分布，就是名称为`y`的那一列：

```
> h2o.table(bank$y)
   y Count
1 no  4000
2 yes  521
[2 rows x 2 columns]
```

可以看到，有521名客户对银行的营销活动给出了“是”的反应，另外4000名客户的反应则是“否”。这个响应变量有点不平衡。后面讲解多类学习时，我们会讨论用于处理不平衡的标号响应变量的技术。在这次练习中，我们看看深度学习技术在这种标号不平衡的情况下表现如何。

7.6.3 建立训练数据集和测试数据集

可以利用H2O平台内置的函数将数据划分为训练集和测试集。首先要为数据建立一个统一的随机数向量：

```
> rand <- h2o.runif(bank, seed = 123)
```

然后即可划分数据，并将数据分配给相应的对象。此时需要指定关键字key的值，如下所示：

```
> train <- bank[rand <= 0.7, ]
> train <- h2o.assign(train, key = "train")
> test <- bank[rand > 0.7, ]
> test <- h2o.assign(test, key = "test")
```

建立训练集和测试集之后，应该看看二者之间的响应变量分布是否均衡。可以使用h2o.table()函数完成这个任务，在我们的例子中，响应变量在第64列：

```
> h2o.table(train[, 64])
    y Count
1  no  2783
2 yes   396
[2 rows x 2 columns]

> h2o.table(test[, 64])
    y Count
1  no  1217
2 yes   125
[2 rows x 2 columns]
```

一切都进行得相当顺利，下面开始构建模型。

7.6.4　模型构建

我们将会看到，在深度学习的建模函数中，需要调优的参数非常少。对于这个平台，我最喜欢的一点就是它使建模过程尽可能简单，只要按照默认设置去做即可。如果你想知道所有默认设置的作用，可以查看帮助文件，也可以使用下面的命令：

```
> args(h2o.deeplearning)
```

可以在线查看对所有参数和参数调优方式的说明文档，网址为http://h2o.ai/docs/master/model/deep-learning/。

顺便说一句，你可以通过demo("method")函数运行各种机器学习方法的演示程序。例如，可以通过demo(h2o.deeplearning)查看深度学习方法的演示。

下一个目标是，使用随机搜索的方法调整超参数，这种方法比全网格搜索要节省时间。我们要检查的超参数有：有舍弃（dropout）和无舍弃的tanh激活函数、3种不同形式的隐藏层（神经元组合）、两种不同的舍弃率，以及两种不同的学习率。

```
> hyper_params <- list(
   activation = c("Tanh", "TanhWithDropout"),
   hidden = list(c(20,20),c(30, 30),c(30, 30, 30)),
   input_dropout_ratio = c(0, 0.05),
```

```
    rate = c(0.01, 0.25)
)
```

你可以将随机搜索原则设置在一个列表里。因为我们要使用随机搜索，所以要将strategy设置为RandomDiscrete；如果要进行全网格搜索，就要设置为Cartesian。我建议你为随机搜索设置一个或多个提前结束标准，比如max_runtime_secs、max_models等。我还设置了一个结束标准，即前5个模型之间的误差在1%以内：

```
> search_criteria = list(
    strategy = "RandomDiscrete", max_runtime_secs = 420,
    max_models = 100, seed = 123, stopping_rounds = 5,
    stopping_tolerance = 0.01
)
```

下面就是h2o.grid()函数大显身手的时候了。我们需要告诉这个函数使用深度学习算法，还有要使用的训练数据集、验证数据（测试数据集）、输入特征和响应变量：

```
    > randomSearch <- h2o.grid(
      algorithm = "deeplearning",
      grid_id = "randomSearch",
      training_frame = train,
      validation_frame = test,
      x = 1:63,
      y = 64,
      epochs = 1,
      stopping_metric = "misclassification",
      hyper_params = hyper_params,
      search_criteria = search_criteria
    )
|===================================================================| 100%
```

有一个进度条会表示函数运行的进程，对于这个数据集，运行时间不过几秒。

检查一下效果最好的前5个模型的结果：

```
> grid <- h2o.getGrid("randomSearch",sort_by = "auc", decreasing =
    FALSE)

> grid
H2O Grid Details
================

Grid ID: randomSearch
Used hyper parameters:
 - activation
 - hidden
 - input_dropout_ratio
 - rate
Number of models: 71
Number of failed models: 0
```

7

```
Hyper-Parameter Search Summary: ordered by decreasing auc
       activation       hidden input_dropout_ratio rate
1 TanhWithDropout  [30, 30, 30]               0.05 0.25
2 TanhWithDropout  [20, 20]                   0.05 0.01
3 TanhWithDropout  [30, 30, 30]               0.05 0.25
4 TanhWithDropout  [40, 40]                   0.05 0.01
5 TanhWithDropout  [30, 30, 30]               0.0  0.25
              model_ids                 auc
1 randomSearch_model_57   0.8636778964667214
2 randomSearch_model_8    0.8623894823336072
3 randomSearch_model_10   0.856568611339359
4 randomSearch_model_39   0.8565258833196385
5 randomSearch_model_3    0.8544026294165982
```

所以，第57号模型最终胜出，它使用有舍弃的tanh激活函数、3个隐藏层（每个隐藏层中有30个神经元）、0.05的舍弃率和0.25的学习率，其AUC大概是0.864。

下面通过混淆矩阵查看模型在测试数据上的错误率：

```
> best_model <- h2o.getModel(grid@model_ids[[1]])
> h2o.confusionMatrix(best_model, valid = T)
Confusion Matrix (vertical: actual; across: predicted) for max f1 @
  threshold = 0.0953170555399435:
         no yes    Error       Rate
no     1128  89 0.073131 = 89/1217
yes      60  65 0.480000 = 60/125
Totals 1188 154 0.111028 = 149/1342
```

尽管错误率只有11%，但yes标签上的错误太多了，它的假阳性率和假阴性率都非常高。这说明不平衡的分类可能是一个问题。我们刚刚开始调整超参数的过程，想改善模型结果还有很多工作要做。这个任务就交给你了！

下面研究如何使用交叉验证建立模型。请注意超参数是如何包含在`h2o.deeplearning()`函数中的，只有学习率是个例外，它被设置为自适应的。我还加入了对少数类进行上采样的设置，以使训练过程中的标号更加平衡。另一个需要注意的是，每折数据都是根据响应变量进行分层抽样得出的：

```
> dlmodel <- h2o.deeplearning(
    x = 1:63,
    y = 64,
   training_frame = train,
   hidden = c(30, 30, 30),
   epochs = 3,
   nfolds = 5,
   fold_assignment = "Stratified",
   balance_classes = T,
   activation = "TanhWithDropout",
   seed = 123,
   adaptive_rate = F,
   input_dropout_ratio = 0.05,
```

```
    stopping_metric = "misclassification",
    variable_importances = T
)
```

如果调用了dlmodel对象，就会得到一个非常长的输出结果。在这个例子中，我们只检查模型在保留折上的效果：

```
> dlmodel
Model Details:
==============
AUC: 0.8571054599
Gini: 0.7142109198

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal
  threshold:
          no yes    Error       Rate
no      2492 291 0.104563 = 291/2783
yes      160 236 0.404040 = 160/396
Totals  2652 527 0.141869 = 451/3179
```

从这个结果可以看出，对超参数的调整任重而道远，特别是对隐藏层数（神经元数）的调整。对模型在样本外数据上的效果的检查过程和前面有些区别，但更全面，使用的是h2o.perfor mance函数：

7

```
> perf <- h2o.performance(dlmodel, test)
> perf
H2OBinomialMetrics: deeplearning
MSE:                   0.07237450145
RMSE:                  0.2690250945
LogLoss:               0.2399027004
Mean Per-Class Error: 0.2326113394
AUC:                   0.8319605588
Gini:                  0.6639211175

Confusion Matrix (vertical: actual; across: predicted) for F1-
  optimal
threshold:
         no yes     Error      Rate
    no 1050 167  0.137223 = 167/1217
   yes   41  84  0.328000 =  41/125
Totals 1091 251  0.154993 = 208/1342

Maximum Metrics: Maximum metrics at their respective thresholds
  metric                      threshold    value idx
1 max f1                       0.323529 0.446809  62
2 max f2                       0.297121 0.612245 166
3 max f0point5                 0.323529 0.372011  62
4 max accuracy                 0.342544 0.906110   0
5 max precision                0.323529 0.334661  62
6 max recall                   0.013764 1.000000 355
7 max specificity              0.342544 0.999178   0
8 max absolute_mcc             0.297121 0.411468 166
```

```
9 max min_per_class_accuracy   0.313356 0.799507 131
10 max mean_per_class_accuracy 0.285007 0.819730 176
```

整体错误率提高了，但假阳性率和假阴性率有所下降。如前所述，需要更多调优工作。

最后，可以计算变量重要性，该操作基于**戈登方法**。请注意，这些结果可能会误导你。在表中，我们看到变量是按照重要性顺序排列的，但变量重要性会受到抽样变动的影响。如果你换一个随机数种子，变量重要性的顺序也很可能发生改变。以下是按重要性排列的前5个变量：

```
> dlmodel@model$variable_importances
Variable Importances:
           variable relative_importance scaled_importance percentage
1 duration                    1.000000          1.000000   0.147006
2 poutcome_success            0.806309          0.806309   0.118532
3 month_oct                   0.329299          0.329299   0.048409
4 month_mar                   0.223847          0.223847   0.032907
5 poutcome_failure            0.199272          0.199272   0.029294
```

综上，借助于H2O平台的功能，我们介绍了如何使用R进行深度学习。H2O平台既简单易用又非常灵活，你可以调整超参数，从而得到最优的模型拟合。好好利用它吧！

7.7　小结

本章目的在于，带领你进入神经网络和深度学习的奇妙世界。我们通过神经网络在两个不同数据集上的实际应用，研究了其工作原理和优缺点。数据中存在错综复杂的非线性关系时，这种技术的效果非常好。但这种技术具有高度的复杂性，可能要调整很多超参数，是个典型的黑盒，非常难以解释。我们不知道为什么自动驾驶汽车会在遇到红灯时向右转，只知道它做得非常妥当。我希望你能充分利用这种技术，或者独立使用，或者以集成建模的方式使用其他技术作为补充。祝你好运且收获多多！下面转入对无监督学习的研究，从聚类开始。

第 8 章

聚类分析

“快给我一杯酒，把我的思想淋湿，我就能说出名言警句了。”

——阿里斯托芬，雅典剧作家

在前面的章节中，我们的重点在于使用最优算法找出一个结果或响应，例如一套乳腺癌诊断方法或一个前列腺特异性抗原水平。这些案例中有一个响应变量Y，Y是X的函数，或表示为$y = f(x)$。我们的数据中有Y的实际值，以Y为依据训练X。这种方式称为**监督式学习**。但是，我们从数据中学习知识的时候通常没有响应变量Y，或者故意忽略了Y。此时就进入**无监督学习**的世界。在这个世界里，我们建立和选择算法的基准是算法能够明确满足业务需求的程度，而不是算法的正确率。

为什么要研究无监督学习？首先，无监督学习可以帮助你理解并识别出数据中的模式，这可能非常有价值。其次，你可以通过无监督学习转换数据，提高监督式学习技术的效果。

本章着重讨论如何识别数据中的模式，下一章讨论数据转换的问题。

那么，我们就从一种广为人知而又功能强大的技术开始——**聚类分析**。聚类分析的目标是将观测分成若干组（k个组），使同一组内的成员尽可能相似，不同组间的成员尽可能不同。聚类分析有很多实际应用，下面仅是其中的一部分例子：

- 客户分类与市场细分
- 高犯罪率地理区域检测
- 图像与人脸识别
- 基因测序与转录
- 石油与地质勘探

聚类分析用途广泛，聚类技术也林林总总。我们将重点讨论两种最常用的技术：**层次**聚类和**K均值**聚类。它们都是非常有效的聚类方法，但如果你要分析的是大规模的高复杂度的数据集，它们可能不太适合。所以，我们还要研究**围绕中心点的划分**，它使用基于**果瓦度量**的相异度矩阵

作为输入。最后讨论一种我最近才学习和使用的新技术——使用随机森林转换数据。转换后的数据可以用作无监督学习的输入。

开始本章内容之前，我们先来看一条评论。你可能会被问到这个问题：既然学习是非监督式的，那么这些技术是不是更接近于艺术，而不是科学？我认为明确的回答是“看情况”。2016年初，印第安纳州印第安纳波利斯市举行了一个R用户小组会议，在这个会议上，我介绍了本书中的一些方法。与会的所有人都赞成这个观点：正是分析师和商业用户的判断才使得无监督学习具有实际意义，并且能够决定在最终的算法中保留3个簇还是4个簇。下面引用的评论很好地总结了这一点：

> “主要障碍在于，不考虑前因后果的时候，很难对聚类算法进行评价。为什么用户从一开始就聚集数据？他们聚集数据之后打算干什么？我们坚信聚类不应该被看作与应用无关的数学问题，而应该一直在其最终用途的背景下展开研究。”
>
> ——拉克斯伯格 等（2012）

8.1　层次聚类

层次聚类算法的基础是观测之间的相异度测量。我们使用的是通用的测量方式——欧氏距离，当然还有其他方式。

> 层次聚类是一种凝聚式的或自底向上的技术。首先，所有观测都是自己本身的一个簇；然后，算法开始在所有的两点组合之中进行迭代搜索，找出最相似的两个簇，将它们聚集成一个簇。所以，第一次迭代之后，有$n-1$个簇；第二次迭代之后，有$n-2$个簇，依此类推。

进行迭代时，除了距离测量之外，还有一个重要问题是需要确定观测组之间的测距方式，不同类型的数据集需要使用不同的簇间测距方式。当你使用不同的测距方式进行实验时，会发现有些测距方式会使一个或几个簇中的观测数量特别不均衡。假设你有30个观测，某种技术可能会一直建立只有一个观测的簇，而不管你想建立多少个。在这种情况下，你必须根据业务和数据的实际情况做出具体的判断，选择最合适的测距方式。

下表列出了常用的测距方式类型，当然还有其他类型的测距方式。

测距方式	描　　述
Ward距离	使总的簇内方差最小，使用簇中的点到质心的误差平方和作为测量方式
最大距离（Complete linkage）	两个簇之间的距离就是两个簇中的观测之间的最大距离
最小距离（Single linkage）	两个簇之间的距离就是两个簇中的观测之间的最小距离
平均距离（Average linkage）	两个簇之间的距离就是两个簇中的观测之间的平均距离
质心距离（Centroid linkage）	两个簇之间的距离就是两个簇的质心之间的距离

层次聚类的输出是一种树状图，可以显示不同簇之间的排列关系。

我们将会看到，选择簇的数目时很难找到一个明确的平衡点。你的决策从本质上说是一个迭代的过程，而且要侧重于业务决策这个大背景。

距离的计算

我们之前提到过，层次聚类一般使用欧氏距离作为输入。下面通过一个简单的例子看看如何计算两个观测之间的欧氏距离，这两个观测都具有两个变量（特征）。

假设观测A的价格为5美元，重量为3磅；观测B的价格为3美元，重量为5磅。将这些值代入距离公式：A和B之间的距离等于它们的特征的差的平方和的平方根。在本例中计算如下：

$$d(A, B) = \text{平方根}((5-3)^2 + (3-5)^2) = 2.83$$

2.83这个值本身没有什么意义，在与其他两点之间距离相比时才有意义。这就是R中的`dist()`函数计算距离的默认方式。你可以在这个函数中指定其他距离计算方式（如最大值距离、曼哈顿距离、堪培拉距离、二值距离和闵可夫斯基距离）。我们不详细讨论你应该在何种情况下，或基于何种原因使用这些距离，以代替欧氏距离，这太专业了。举例来说，你的数据受到维度过高的影响时，欧氏距离就不适合了，比如基因组研究。你需要使用专业领域知识和反复实验来确定合适的距离测量方式。

> 最后需要注意的是，要对你的数据进行标准化的缩放操作，使数据的均值为0，标准差为1，这样在计算距离时才能进行比较。否则，变量的测量值越大，对距离的影响就越大。

8.2　K 均值聚类

使用K均值聚类时，需要明确指定所需的簇的数目，然后算法开始迭代，直到每个观测都属于某个簇。算法的目标是使簇内的差异最小，簇内差异由欧氏距离的平方定义。所以，第*k*个簇的簇内差异等于簇内所有两个观测之间的欧氏距离的平方和，再除以簇内观测的数量。

因为迭代过程的存在，一次K均值聚类的结果可能和另一次的结果大相径庭——尽管你指定了同样数目的簇。看看这是如何发生的。

(1) **设定**：你需要的簇的确切数量（*k*）。

(2) **初始化**：随机选择*k*个观测作为初始均值。

(3) **迭代**：

- 将每个观测分配给离它最近的簇中心点（使簇内平方和最小），建立*k*个簇；
- 将每个簇的中心点作为新的均值；

■ 重复上面两个步骤，直至收敛，即簇中心点不再改变。

可以看出，因为第1步中的初始分配是随机的，所以会造成每次聚类结果不一致。因此，重要的一点是，要进行多次初始分配，让软件找出最优的解。在R中，你可以看到这个过程非常容易。

8.3　果瓦系数与围绕中心的划分

你开始实际的聚类分析时，很快就会遇到一个问题：无论层次聚类还是K均值聚类，都不是为分析混合数据而专门设计的。什么是混合数据呢？就是既包括定量数据又包括定性数据的数据，说得更具体一些，就是包括名义数据、定序数据、定距数据和定比数据。

现实情况就是，你使用的大多数数据集都包括混合数据。有很多方式可以处理混合数据，比如可以先进行**主成分分析**（第9章），建立潜变量，然后使用潜变量作为聚类的输入，还可以使用不同的相异度计算方式。

在简单易用而又功能强大的R中，你可以使用**果瓦相异度系数**将混合数据转换为适当的特征空间。在这种方法中，你甚至可以使用因子作为输入变量。还有，在聚类算法方面，我推荐使用**PAM聚类算法**，而不是K均值。

PAM和K均值很相似，但是有两个明显的优点：

- 第一，PAM可以接受相异度矩阵作为输入，这样即可处理混合数据；
- 第二，PAM对于异常值和不对称数据的鲁棒性更好，因为它最小化的是相异度总和，而不是欧氏距离的平方和（Reynolds，1992）。

这并不是说必须同时使用果瓦系数和PAM，如果你愿意，完全可以在层次聚类方法中使用果瓦系数。对于K均值方法，我也见到过对果瓦系数的意见，赞成和反对都有。除此以外，PAM还能接受其他测距方式。PAM和果瓦系数构成了处理混合数据的一套有效方法。我们先简单介绍这两个概念，然后讨论其他内容。

8.3.1　果瓦系数

果瓦系数比较两个成对的实例，并计算它们之间的相异度，实质上就是每个变量的贡献的加权平均值。对于两个实例i与j，果瓦系数定义如下：

$$S_{ij} = \mathrm{sum}(W_{ijk} * S_{ijk}) / \mathrm{sum}(W_{ijk})$$

其中，S_{ijk}是第k个变量的贡献。如果第k个变量是有效的，W_{ijk}是1，否则是0。

对于定序变量和连续变量，$S_{ijk} = 1 - (x_{ij} - x_{ik}$的绝对值$)/r_k$，$r_k$是第$k$个变量的取值范围。

对于名义变量，如果$x_{ij} = x_{jk}$，那么$S_{ijk} = 1$，否则为0。

对于二值变量，S_{ijk}是基于具有(+)或不具有（–）某个属性来计算的，如下表所示。

变　量	第k个属性的值			
实例i	+	+	–	–
实例j	+	–	+	–
S_{ijk}	1	0	0	0
W_{ijk}	1	1	1	0

8.3.2 PAM

对于**围绕中心点的划分**，我们先定义**中心点**。

> 中心点是簇内所有观测中，使相异度（使用果瓦系数表示）最小的那个观测。所以，同K均值一样，如果指定5个簇，就可以将数据划分为5份。

PAM算法的目标是，使所有观测与离它们最近的中心点的相异度最小。该算法按照下面的步骤迭代：

(1) 随机选择k个观测作为初始中心点；
(2) 将每个观测分配至最近的中心点；
(3) 用非中心点观测替换中心点，并计算相异度的变化；
(4) 选择能使总相异度最小的配置；
(5) 重复第(2)步~第(4)步，直至中心点不再变化。

果瓦系数和PAM都可以使用R中的cluster包实现。使用`daisy()`函数计算相异度矩阵，从而计算果瓦系数，然后使用`pam()`函数进行实际的数据划分。下面开始实验这些方法。

8.4 随机森林

出于和使用果瓦度量处理混合（说实话，是混乱的）数据同样的考虑，也可以以非监督的方式使用随机森林。选择这种方法有以下优点：

- 对异常值和高度扭曲的变量的鲁棒性更好；
- 不需要对数据进行转换和比例缩放；
- 可以处理混合数据（数值型和因子）；
- 可以兼容缺失值；
- 可以在有大量变量的数据上使用，实际上，通过检查变量重要性，可以消除那些无用的特征；

- 生成的相异度矩阵可以作为我们前面讨论过的一些技术的输入（层次聚类、K均值聚类和PAM）。

我要给出一些忠告。这种方法会经过多次试错来对随机森林进行调优，比如需要不断调整每次树分裂时变量抽样的数目（函数里参数`mtry`的值），以及森林中树的数目。研究表明，在一定程度上，树的数目越多，结果就越好，2000棵树是一个很好的起点（Shi, T. 与Horvath, S.，2006）。

以下是算法的基本步骤，适用于无标号的数据集：

- 将当前观测数据标记为类别1；
- 创建另一个与上面的观测数据同样大小的（合成）集合，创建方法是使用观测数据中的每个特征进行随机抽样，所以如果观测数据中有20个特征，那么合成数据中也有20个同样的特征；
- 将合成集合中的数据标记为类别2，这样就人工创建了一个适合应用随机森林的分类问题；
- 创建一个随机森林模型，使用上面的两个类别对数据进行分类；
- 将模型在观测数据（丢弃合成数据）上的邻近度转换为相异度矩阵；
- 使用这个相异度矩阵作为聚类的输入特征。

那么，什么是邻近度呢？

邻近度是所有观测两两之间的一种度量。如果两个观测能够到达一棵树的同一个终端节点，它们的邻近度就是1，否则是0。

随机森林模型运行结束后，要对观测数据的邻近度分数进行标准化，除以树的总数即可。最终的$N \times N$矩阵中，分数值范围为0～1，当然，对角线上都是1。这种方法就介绍到这里，它非常有效，但尚未得到广泛应用，我要是早点学会这种方法就好了。

8.5　业务理解

直到几周前我才知道，全球只有不到300位经过认证的侍酒大师。世界侍酒大师协会主持的认证考试以要求严苛、失败率高而名闻遐迩。

纪录片《侍酒师》广受好评，详细纪录了几位年轻人在追求侍酒大师认证过程中的努力拼搏，以及获得的丰厚回报。下面进行实战练习，帮助一位梦想成为侍酒大师而奋斗的年轻人发现意大利葡萄酒的潜在结构。

8.6　数据理解与数据准备

首先加载本章所需的R包，和往常一样，你要确定已经安装了这些R包：

```
> library(cluster) #conduct cluster analysis
> library(compareGroups) #build descriptive statistic tables
> library(HDclassif) #contains the dataset
> library(NbClust) #cluster validity measures
> library(sparcl) #colored dendrogram
```

数据集位于HDclassif包中，我们已经安装好了。加载数据，使用`str()`函数检查数据结构：

```
> data(wine)

> str(wine)
'data.frame':178 obs. of  14 variables:
 $ class: int  1 1 1 1 1 1 1 1 1 1 ...
 $ V1   : num  14.2 13.2 13.2 14.4 13.2 ...
 $ V2   : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35
      ...
 $ V3   : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...
 $ V4   : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...
 $ V5   : int  127 100 101 113 118 112 96 121 97 98 ...
 $ V6   : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...
 $ V7   : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15
      ...
 $ V8   : num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...
 $ V9   : num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85
      ...
 $ V10  : num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...
 $ V11  : num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01
      ...
 $ V12  : num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...
 $ V13  : int  1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...
```

数据包括178种葡萄酒，有13个变量表示酒中的化学成分，还有一个标号变量Class，表示品种等级或葡萄种植品种。在聚类过程中，我们不会使用这个标号变量，而是用它验证模型性能。变量`V1`~`V13`表示酒中化学成分的测量结果，如下所示。

- `V1`：酒精
- `V2`：苹果酸
- `V3`：灰分
- `V4`：灰分碱性
- `V5`：镁
- `V6`：总酚含量
- `V7`：黄酮类化合物
- `V8`：非黄酮类酚类
- `V9`：原花青素
- `V10`：色彩强度
- `V11`：色调
- `V12`：OD280/OD315

- V13：脯氨酸

变量都是定量的，应该将变量重命名为对于我们的分析有意义的形式，使用name()函数即可：

```
    > names(wine) <- c("Class", "Alcohol", "MalicAcid", "Ash",
      "Alk_ash", "magnesium", "T_phenols", "Flavanoids", "Non_flav",
        "Proantho", "C_Intensity", "Hue", "OD280_315", "Proline")

> names(wine)
 [1] "Class"     "Alcohol"   "MalicAcid"   "Ash"
 [5] "Alk_ash"   "magnesium" "T_phenols"   "Flavanoids"
 [9] "Non_flav"  "Proantho"  "C_Intensity" "Hue"
[13] "OD280_315" "Proline"
```

因为变量没有进行标准化，所以使用scale()函数完成这个工作。这个函数先将数据中心化，即用每列的变量值减去这一列的均值。中心化的数据再除以相应列的标准差即可完成标准化。转换数据时，要确认只对第2列~第14列数据操作，跳过Class列，转换后的数据保存在一个数据框中。以上操作都可以通过一行代码完成：

```
> df <- as.data.frame(scale(wine[, -1]))
```

现在检查数据结构，确保以上操作都按计划进行：

```
    > str(df)
'data.frame':178 obs. of 13 variables:
 $ Alcohol    : num  1.514 0.246 0.196 1.687 0.295 ...
 $ MalicAcid  : num  -0.5607 -0.498 0.0212 -0.3458 0.2271 ...
 $ Ash        : num  0.231 -0.826 1.106 0.487 1.835 ...
 $ Alk_ash    : num  -1.166 -2.484 -0.268 -0.807 0.451 ...
 $ magnesium  : num  1.9085 0.0181 0.0881 0.9283 1.2784 ...
 $ T_phenols  : num  0.807 0.567 0.807 2.484 0.807 ...
 $ Flavanoids : num  1.032 0.732 1.212 1.462 0.661 ...
 $ Non_flav   : num  -0.658 -0.818 -0.497 -0.979 0.226 ...
 $ Proantho   : num  1.221 -0.543 2.13 1.029 0.4 ...
 $ C_Intensity: num  0.251 -0.292 0.268 1.183 -0.318 ...
 $ Hue        : num  0.361 0.405 0.317 -0.426 0.361 ...
 $ OD280_315  : num  1.843 1.11 0.786 1.181 0.448 ...
 $ Proline    : num  1.0102 0.9625 1.3912 2.328 -0.0378 ...
```

进行下一步之前，通过一个表格看看品种等级Class的分布：

```
    > table(wine$Class)

 1  2  3
59 71 48
```

下面进入模型构建阶段。

8.7　模型构建与模型评价

建立数据框df之后，可以使用聚类算法。先使用层次聚类，然后再试试K均值方法。此后需要对数据进行一些调整，演示如何使用果瓦系数和随机森林处理混合数据。

8.7.1　层次聚类

要在R中建立层次聚类模型，可以使用stats包中的`hclust()`函数。这个函数需要两个基本输入：距离矩阵和聚类方法。使用`dist()`函数可以轻松生成距离矩阵，我们使用的是欧氏距离。可以使用的聚类方法有若干种，`hclust()`函数使用的默认方法是最大距离法。

我们将使用默认方法，但我建议你试试Ward距离法，这种方法得出的簇的观测数量相差可能更小。

在最大距离法得出的结果中，两个簇之间的距离等于两个簇中的观测之间的最大距离。Ward距离法对观测进行聚集的目标是，使簇内观测的误差平方和最小。

值得注意的是，在R中，ward.D2方法使用欧氏距离的平方来测量距离，这是真正的Ward距离法。R中还有一个ward.D方法，但它要求使用距离矩阵的平方作为输入值，因为我们建立距离矩阵时用的是未进行平方运算的数值，所以需要使用ward.D2。

现在的一个重要问题是，应该生成多少个簇？就像本章开始部分所说的，答案就是"看情况"，但这个答案太简单了，不那么令人满意。我们将会看到，要解决这个难题，除了需要一些聚类有效性指标以外，确实还需要对商业环境和基础数据的深刻理解，以及不断的测试和纠错。因为我们的侍酒师是虚构的，所以只能依靠聚类有效性指标。但是，在选择簇的数目方面没有万能良药，因为有效性指标多达几十种。

对大量聚类有效性指标的优缺点的研究已经超出了本章的范围，既然如此，我们就通过两篇论文以及R语言本身来简化这个问题。Miligan和Cooper在1985年的一篇论文中，使用模拟数据研究了30种不同的聚类有效性指标。表现最好的前5种指标是CH指数、Duda指数、Cindex、Gamma和Beale指数。另外一种确定簇数目的著名方法是gap统计量（Tibshirani、Walther与Hastie，2001）。这两篇论文可以帮助你选择合适的聚类有效性指标。

在R中，我们可以使用NbClust包中的`NbClust()`函数，求出23种聚类有效性指标的结果，包括Miligan和Cooper论文中最好的5种和gap统计量。你可以在这个R包的帮助文件中找到所有指标的列表。有两种方式可以执行这个函数，一种是选择你最喜欢的指标，通过R语言进行调用；另一种方式是在分析时包括所有指标，然后按照少数服从多数的原则进行选择。这个函数可以生成一个非常详细的摘要，同时还可以生成两张统计图。

说完这个问题之后，尝试通过示例使用最大距离法。使用这个函数时，你需要指定簇的最小

值和最大值、距离类型、测距方式和有效性指标。在以下的代码中可以看到，我们要建立一个名为numComplete的对象，函数指定使用欧氏距离，簇的最小数量为2，最大数量为6，测距方式为最大距离法，并使用所有有效性指标。函数运行后会自动输出类似下面的内容——既使用图形方法，又使用少数服从多数原则来确定簇的数目：

```
    > numComplete <- NbClust(df, distance = "euclidean", min.nc = 2,
       max.nc=6, method = "complete", index = "all")
*** : The Hubert index is a graphical method of determining the
       number of clusters.
In the plot of Hubert index, we seek a significant knee that
       corresponds to a significant increase of the value of the
         measure that is the significant peak in Hubert index second
           differences plot.

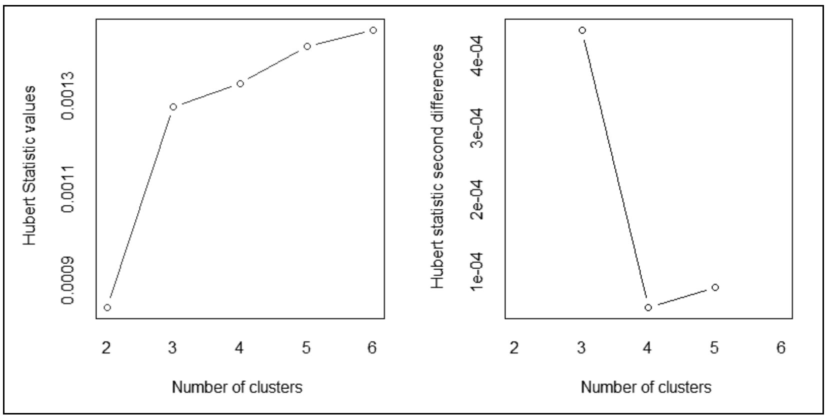
*** : The D index is a graphical method of determining the number
       of clusters.
In the plot of D index, we seek a significant knee (the significant peak in
Dindex second differences plot) that corresponds to a significant increase
of the value of the measure.

*******************************************************************
* Among all indices:
* 1 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 6 proposed 5 as the best number of clusters
* 5 proposed 6 as the best number of clusters

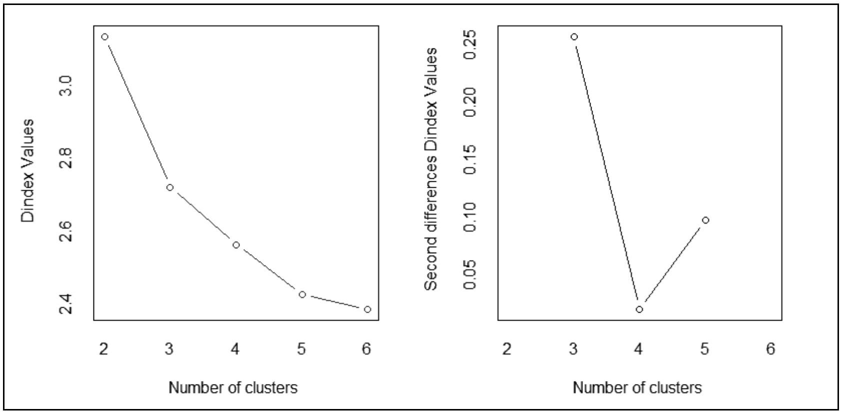
                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is 3
```

根据少数服从多数原则，我们应该选择3个簇作为最优解，至少对于层次聚类是这样。对于生成的两张统计图，每张图中有两幅图。正如前面的输出结果所示，在左侧的图中，你要找出一个明显的拐点；在右侧的图中，你要找到峰值。下面是**Hubert指数**图。



可以看到，左图在3个簇的地方有个拐点，右图在3个簇的时候达到峰值。下面的**Dindex图**也提供了同样的信息。



`NbClust()`函数的结果中有很多种数值，我只介绍其中一种——每种有效性指标的最优簇数量和与之对应的指标值。这些数值可以通过$Best.nc引用。我只列出前9个指标，省略其他：

```
> numComplete$Best.nc
                     KL      CH Hartigan   CCC    Scott
Number_clusters  5.0000 3.0000    3.0000 5.000   3.0000
Value_Index     14.2227 48.9898  27.8971 1.148 340.9634
                     Marriot   TrCovW   TraceW Friedman
Number_clusters 3.000000e+00     3.00   3.0000   3.0000
Value_Index     6.872632e+25 22389.83 256.4861  10.6941
```

可以看到第一个指标（KL）的最优簇数量是5，第二个指标（CH）的最优簇数量是3。

选择3个簇进行聚类，现在计算距离矩阵，并建立层次聚类模型。如下所示：

```
> dis <- dist(df, method = "euclidean")
```

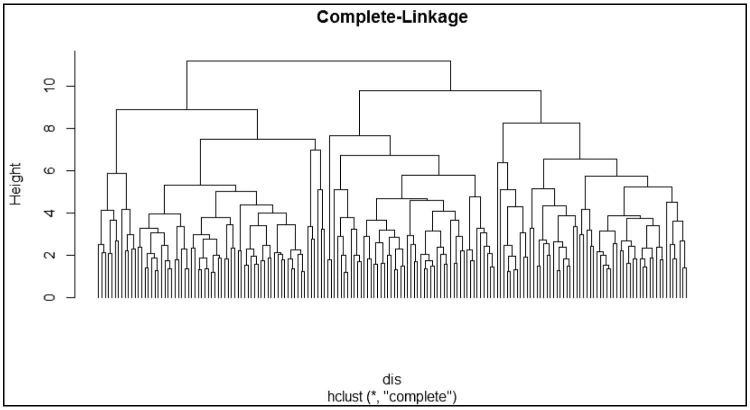
然后，使用这个矩阵作为`hclust()`函数的输入，进行实际的聚类过程：

```
> hc <- hclust(dis, method = "complete")
```

层次聚类可视化的通用方式是画出**树状图**，可以用plot函数实现。注意，参数`hang=-1`表示将观测排列在图的底部：

```
> plot(hc, hang = -1, labels = FALSE, main = "Complete-Linkage")
```

这个树状图表明了观测是如何聚集的，图中的连接（也可称为分支）告诉我们哪些观测是相似的。分支的高度表示观测之间相似或相异的程度，这个程度可以从距离矩阵中得到。请注意，我设置了参数`labels=FALSE`，这是为了使统计图清晰可读，因为观测数量太多。如果是一个不多于40个观测的小数据集，则可以显示每行标记。

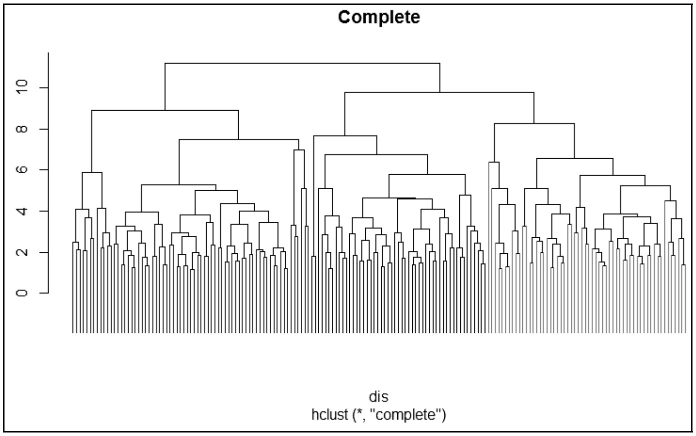


如果想使聚类可视化效果更好，可以使用sparcl包生成彩色树状图。要对合适数目的簇上色，需要使用cutree()函数对树状图进行剪枝，以得到合适的簇的数目。这个函数还可以为每个观测生成簇标号：

```
> comp3 <- cutree(hc, 3)
```

可以在函数中使用comp3对象生成彩色树状图：

```
> ColorDendrogram(hc, y = comp3, main = "Complete", branchlength = 50)
```



请注意，我指定了参数branchlength=50，这个值要根据你自己的数据确定。因为我们已经有了簇标号，所以可以建立一个表格查看每个簇中观测的数量：

```
    > table(comp3)
comp3
  1  2  3
 69 58 51
```

出于好奇，我们比较这个聚类结果和品种等级的标号：

```
> table(comp3,wine$Class)

comp3  1  2  3
    1 51 18  0
    2  8 50  0
    3  0  3 48
```

这个表中，行是簇标号，列是品种等级标号。聚类结果匹配了84%的品种等级。请注意，我们不想用聚类结果去预测品种等级，在这个例子中没有任何理由这样做。

下面试试Ward距离法。代码和前面的一样，首先确定簇的数目，应该将method的值改为`Ward.D2`：

```
> numWard <- NbClust(df, diss = NULL, distance = "euclidean",
  min.nc = 2, max.nc = 6, method = "ward.D2", index = "all")

*** : The Hubert index is a graphical method of determining the number of
clusters.
In the plot of Hubert index, we seek a significant knee that corresponds to
a significant increase of the value of the measure i.e the significant peak
in Hubert index second differences plot.

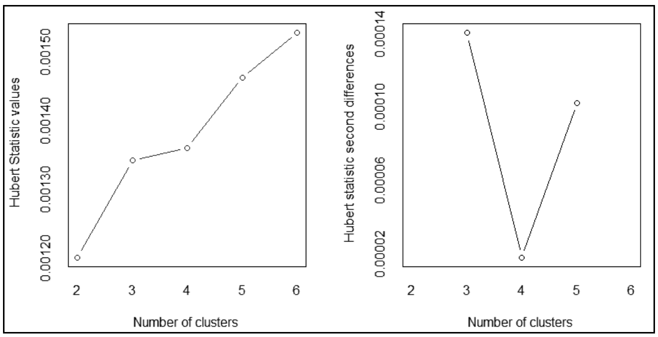
*** : The D index is a graphical method of determining the number of
clusters.
In the plot of D index, we seek a significant knee (the significant peak in
Dindex second differences plot) that corresponds to a significant increase
of the value of the measure.

*******************************************************************
* Among all indices:
* 2 proposed 2 as the best number of clusters
* 18 proposed 3 as the best number of clusters
* 2 proposed 6 as the best number of clusters

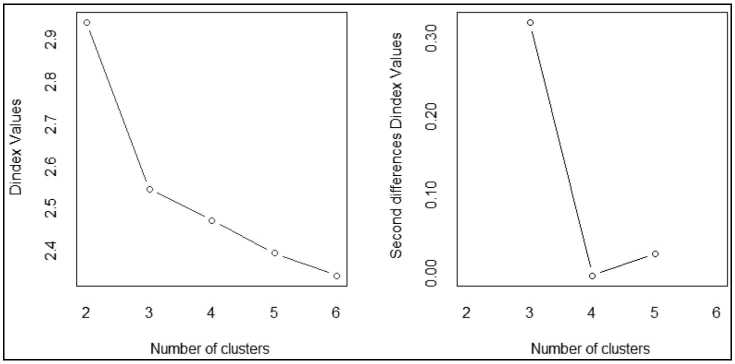
                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is 3
```

这次的结果也没什么变化，根据少数服从多数原则，簇的数目是3。看一下Hubert指数图，最优解也是3个簇。

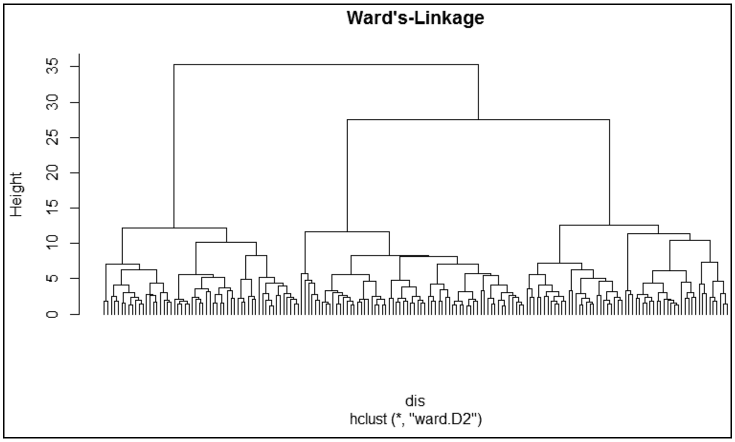


Dindex图提供的最优解也是3个簇。



下面开始实际的聚类过程，生成Ward距离法的树状图：

```
> hcWard <- hclust(dis, method = "ward.D2")
> plot(hcWard, labels = FALSE, main = "Ward's-Linkage")
```



图中显示的3个簇区别十分明显，每个簇中观测的数量大致相同。计算每个簇的大小，并与品种等级标号进行比较：

```
> ward3 <- cutree(hcWard, 3)
> table(ward3, wine$Class)

ward3  1  2  3
    1 59  5  0
    2  0 58  0
    3  0  8 48
```

可以看到，第一个簇中有64个观测，第二个簇中有58个观测，第三个簇中有56个观测。与最大距离法相比，这种方法和品种等级的分类更加匹配。

通过另一个表比较两种方法的观测匹配情况：

```
    > table(comp3, ward3)
     ward3
comp3  1  2  3
    1 53 11  5
    2 11 47  0
    3  0  0 51
```

两种方法中的第三个簇非常接近，其他两个簇则有差别。那么问题来了，我们如何找出这种差别解释这个现象？很多例子中的数据集非常小，你可以查看每个簇中的观测。但在实际世界中，这经常是不可能的。一个比较好的方法是使用aggregate()函数，计算均值或中位数等统计量。此外，计算时不使用标准化后的数据，而是在原始数据上进行实验。在这个函数中，你需要指定数据集、聚集数据的依据和摘要统计量：

```
    > aggregate(wine[, -1], list(comp3), mean)
  Group.1 Alcohol MalicAcid      Ash  Alk_ash magnesium T_phenols
1       1 13.40609 1.898986  2.305797 16.77246 105.00000  2.643913
2       2 12.41517 1.989828  2.381379 21.11724  93.84483  2.424828
3       3 13.11784 3.322157  2.431765 21.33333  99.33333  1.675686
  Flavanoids  Non_flav Proantho C_Intensity       Hue OD280_315  Proline
1  2.6689855 0.2966667 1.832899    4.990725 1.0696522  2.970000  984.6957
2  2.3398276 0.3668966 1.678103    3.280345 1.0579310  2.978448  573.3793
3  0.8105882 0.4443137 1.164314    7.170980 0.6913725  1.709804  622.4902
```

这样就计算出数据中13个变量在各个簇中的均值。最大距离法的计算完成之后，计算Ward距离法：

8

```
    > aggregate(wine[, -1], list(ward3), mean)
  Group.1  Alcohol MalicAcid      Ash  Alk_ash magnesium T_phenols
1       1 13.66922 1.970000  2.463125 17.52812 106.15625  2.850000
2       2 12.20397 1.938966  2.215172 20.20862  92.55172  2.262931
3       3 13.06161 3.166607  2.412857 21.00357  99.85714  1.694286
  Flavanoids  Non_flav Proantho C_Intensity      Hue OD280_315   Proline
1  3.0096875 0.2910937 1.908125    5.450000 1.071406  3.158437 1076.0469
2  2.0881034 0.3553448 1.686552    2.895345 1.060000  2.862241  501.4310
3  0.8478571 0.4494643 1.129286    6.850179 0.721000  1.727321  624.9464
```

各个数值都非常接近。Ward距离法中的第一个簇中所有变量的均值要稍高一些，第二个簇中除了Hue以外，其他变量的均值要稍低一些。可以将这个结果与具有领域专业知识的人分享一下，请求他们给出解释。我们用统计图表示两种方法中的每个簇的变量值，这样有助于理解和解释。

箱线图可以很好地比较变量的分布，它可以展示最小值、第一四分位数、中位数、第三四分位数、最大值和可能的离群点。

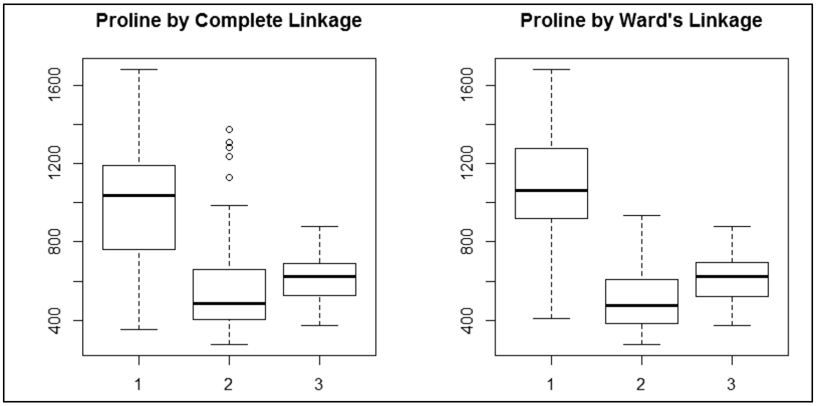
假设我们对两种聚类方法中的Proline值很感兴趣，可以生成一个带有两幅箱线图的统计图进行比较。首先要准备制图区域，使两幅图并列排列。使用par()函数即可：

```
> par(mfrow =c (1, 2))
```

设置参数1mfrow=c(1, 2))表示需要一行两列的统计图，如果需要两行一列的统计图，则应该设定1mfrow=c(2, 1))。使用boxplot()函数时，通过波形符号（~）指定Y轴上的值是X轴上的值的函数：

```
    > boxplot(wine$Proline ~ comp3, data = wine,
              main="Proline by Complete Linkage")

> boxplot(wine$Proline ~ ward3, data = wine,
              main = "Proline by Ward's Linkage")
```



看一下箱线图，其中的方框代表第一四分位数、中位数（方框中间的粗横线）和第三四分位数，称为**四分位距**。虚线的端点（通常称为**须**）代表最大值和最小值。可以看到，最大距离法的第二个簇的最大值上面有5个空心小圆点，这是**疑似离群点**，也就是值超过方框两端1.5倍四分位距的点。

任何值超过方框两端3倍四分位距的点都被认定为离群点，用实心圆点表示。Ward距离法中的第一个和第二个簇具有更紧凑的四分位距，没有疑似离群点。这是我的个人意见，仅供参考。

TIP　查看每个变量的箱线图有助于你和领域专家确定最好的层次聚类方法。讨论完这些问题之后，下面介绍K均值聚类。

8.7.2　K 均值聚类

正如我们在层次聚类中做的那样，可以使用NbClust()函数确定K均值聚类的最优簇数目。只需在函数中将method的值设定为kmeans即可，同时将最大簇数目放大到15。我简化了输出，只列出了少数服从多数原则的部分：

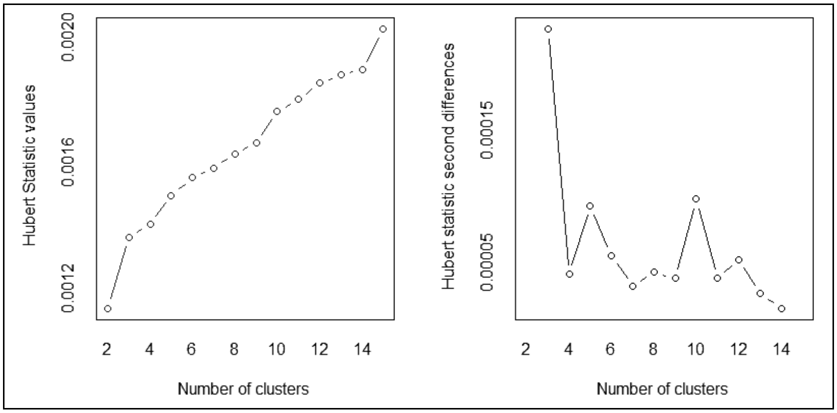
```
    > numKMeans <- NbClust(df, min.nc = 2, max.nc = 15, method =
      "kmeans")
* Among all indices:
```

```
* 4 proposed 2 as the best number of clusters
* 15 proposed 3 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 1 proposed 12 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is 3
```

3个簇再次成为最优解。下面是Hubert图，也证明了这一点。



在R中，我们使用`kmeans()`函数进行K均值聚类。除了输入数据之外，还要指定刚刚得出的簇的数目，以及表示随机分配的参数nstart，还需要设定随机数种子：

```
> set.seed(1234)

> km <- kmeans(df, 3, nstart = 25)
```

建立一个表格，可以知道观测在簇之间的分布情况：

```
> table(km$cluster)

 1  2  3
62 65 51
```

簇之间的观测数量分布得非常均衡。我曾经不止一次遇到过这种情况，在一个有很多变量的大数据集中，不管使用多少个簇的K均值聚类，都得不到有价值且令人信服的结果。对聚类结果的另一种分析方式是查看簇中心点矩阵，它保存了每个簇中每个变量的中心点值：

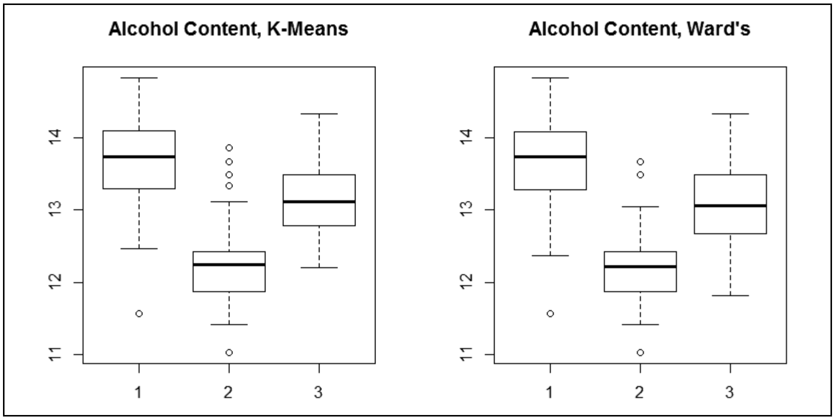
```
> km$centers
    Alcohol MalicAcid        Ash    Alk_ash   magnesium T_phenols
1  0.8328826 -0.3029551  0.3636801 -0.6084749  0.57596208 0.88274724
2 -0.9234669 -0.3929331 -0.4931257  0.1701220 -0.49032869 -0.07576891
3  0.1644436  0.8690954  0.1863726  0.5228924 -0.07526047 -0.97657548
```

```
   Flavanoids    Non_flav    Proantho C_Intensity        Hue   OD280_315
1  0.97506900 -0.56050853  0.57865427   0.1705823  0.4726504  0.7770551
2  0.02075402 -0.03343924  0.05810161  -0.8993770  0.4605046  0.2700025
3 -1.21182921  0.72402116 -0.77751312   0.9388902 -1.1615122 -1.2887761
     Proline
1  1.1220202
2 -0.7517257
3 -0.4059428
```

注意，第一个簇具有相对较高的酒精成分。我们做一个箱线图看看酒精成分的分布，和前面的做法一样，用Ward距离法进行对比：

```
> boxplot(wine$Alcohol ~ km$cluster, data = wine,
            main = "Alcohol Content, K-Means")

> boxplot(wine$Alcohol ~ ward3, data = wine,
            main = "Alcohol Content, Ward's")
```



每个簇中的酒精成分在两幅图中几乎是一模一样的。这在表面上说明了3个簇正是wine数据集中的潜在结构，而且，使用K均值方法和层次聚类方法在结果上没有什么大的区别。最后，对比K均值聚类结果和品种等级：

```
> table(km$cluster, wine$Class)

   1  2  3
1 59  3  0
2  0 65  0
3  0  3 48
```

这与Ward距离法产生的结果非常相似，对于我们虚拟的侍酒师来说，任何一种方法的结果都可以接受。

为了演示如何在既有数值型数据又有非数值型数据的数据集上进行聚类，再介绍另外几个例子。

8.7.3 果瓦系数和 PAM

要开始这个步骤，需要对数据做一点整理。因为这种方法可以处理因子变量，所以可以将酒精函数转换为因子，它有两个水平：高/低。使用ifelse()函数，时需一行代码即可将变量转换为因子。如果酒精含量大于0，因子的水平就是High，否则为Low：

```
> wine$Alcohol <- as.factor(ifelse(df$Alcohol > 0, "High", "Low"))
```

现在建立相异度矩阵，使用cluster包中的daisy()函数，指定方法为gower：

```
> disMatrix <- daisy(wine[, -1], metric = "gower")
```

要建立聚类对象，可以使用cluste包中的pam()函数，将聚类对象命名为pamFit。建立3个簇，并生成一个表格表明簇的大小：

```
    > set.seed(123)

> pamFit <- pam(disMatrix, k = 3)

> table(pamFit$clustering)

 1  2  3
63 67 48
```

然后，对比聚类结果和品种等级：

```
    > table(pamFit$clustering, wine$Class)

         1  2  3
      1 57  6  0
      2  2 64  1
      3  0  1 47
```

接受这个结果，并使用compareGroups包建立一张描述性统计表。在R的基础包中，很难建立可以用于商业展示的表格，compareGroups包则非常好地解决了这个问题。第一步是通过包中的compareGroups()函数建立一个对象，保存聚类结果中的描述性统计。然后使用createTable()函数，将描述性统计转换成一个非常容易导出的表格。我们将以.csv文件的形式将其导出，如果你愿意，可以将表格导出为.pdf、HTML或者LaTeX文件形式：

```
    > wine$cluster <- pamFit$clustering

> group <- compareGroups(cluster ~ ., data = wine)

> clustab <- createTable(group)

> clustab

--------Summary descriptives table by 'cluster'---------

    ____________________________________________________________
```

```
                    1              2             3        p.overall
                  N=63           N=67          N=48
  -----------------------------------------------------------
  Class          1.10 (0.30)   1.99 (0.21)  2.98 (0.14)  <0.001
  Alcohol:                                               <0.001
         High      63 (100%)     1 (1.49%)   28 (58.3%)
          Low      0 (0.00%)    66 (98.5%)   20 (41.7%)
  MalicAcid      1.98 (0.83)    1.92 (0.90) 3.39 (1.05)  <0.001
  Ash            2.42 (0.27)    2.27 (0.31) 2.44 (0.18)   0.001
  Alk_ash        17.2 (2.73)    20.2 (3.28) 21.5 (2.21)  <0.001
  magnesium       105 (11.6)    95.6 (17.2) 98.5 (10.6)   0.001
  T_phenols      2.82 (0.36)    2.24 (0.55) 1.68 (0.36)  <0.001
  Flavanoids     2.94 (0.47)    2.07 (0.70) 0.79 (0.31)  <0.001
  Non_flav       0.29 (0.08)    0.36 (0.12) 0.46 (0.12)  <0.001
  Proantho       1.86 (0.47)    1.64 (0.59) 1.17 (0.41)  <0.001
  C_Intensity    5.41 (1.31)    3.05 (0.89) 7.41 (2.29)  <0.001
  Hue            1.07 (0.13)    1.05 (0.20) 0.68 (0.12)  <0.001
  OD280_315      3.10 (0.39)    2.80 (0.53) 1.70 (0.27)  <0.001
  Proline         1065 (280)     533 (171)   628 (116)   <0.001
  comp_cluster   1.16 (0.37)    1.81 (0.50) 3.00 (0.00)  <0.001
-----------------------------------------------------------
```

这张表显示了因子水平在各个簇中的比例，对于数值型变量，显示了均值以及括号中的标准差。要将表格导出为.csv文件，使用`export2csv()`函数即可：

```
> export2csv(clustab,file = "wine_clusters.csv")
```

打开这个文件可以得到下面的表，这个表对于更深入的分析有指导作用，而且非常适用于商业展示。

	1	2	3	p.overall
	N=60	N=69	N=49	
Alcohol:				<0.001
High	58 (96.7%)	6 (8.70%)	28 (57.1%)	
Low	2 (3.33%)	63 (91.3%)	21 (42.9%)	
MalicAcid	-0.31 (0.62)	-0.37 (0.89)	0.90 (0.97)	<0.001
Ash	0.28 (0.89)	-0.42 (1.14)	0.25 (0.67)	<0.001
Alk_ash	-0.75 (0.76)	0.24 (1.00)	0.58 (0.67)	<0.001
magnesium	0.43 (0.77)	-0.34 (1.18)	-0.05 (0.77)	<0.001
T_phenols	0.87 (0.54)	-0.06 (0.86)	-0.99 (0.56)	<0.001
Flavanoids	0.96 (0.40)	0.04 (0.70)	-1.23 (0.31)	<0.001
Non_flav	-0.58 (0.56)	0.00 (0.98)	0.71 (1.00)	<0.001
Proantho	0.55 (0.72)	0.05 (1.06)	-0.75 (0.72)	<0.001
C_Intensity	0.20 (0.53)	-0.87 (0.38)	0.99 (1.00)	<0.001
Hue	0.46 (0.51)	0.44 (0.89)	-1.19 (0.51)	<0.001
OD280_315	0.77 (0.50)	0.25 (0.69)	-1.30 (0.38)	<0.001
Proline	1.14 (0.74)	-0.72 (0.51)	-0.38 (0.37)	<0.001
comp_cluster	-0.94 (0.42)	-0.14 (0.59)	1.35 (0.00)	<0.001
ward_cluster	-1.16 (0.00)	0.11 (0.49)	1.27 (0.00)	<0.001
km_cluster	-1.16 (0.16)	0.06 (0.34)	1.33 (0.00)	<0.001
class:				<0.001
1	59 (98.3%)	0 (0.00%)	0 (0.00%)	
2	1 (1.67%)	69 (100%)	1 (2.04%)	
3	0 (0.00%)	0 (0.00%)	48 (98.0%)	

最后，使用随机森林创建一个相异度矩阵，并通过PAM方法创建3个簇。

8.7.4　随机森林与 PAM

要在R中执行这种方法，可以使用randomForest()函数。设定随机数种子之后，即可创建模型对象。在下面的代码中，我设定了树的数目为2000，邻近度度量为TRUE：

```
> set.seed(1)

> rf <- randomForest(x = wine[, -1], ntree = 2000, proximity = T)

> rf

Call:
  randomForest(x = wine[, -1], ntree = 2000, proximity = T)
  Type of random forest: unsupervised
  Number of trees: 2000
No. of variables tried at each split: 3
```

可以看到，调用rf对象除了能知道每次树分裂时抽取的变量数（mtry）之外，没有任何有意义的输出。我们先检查这个$N \times N$矩阵的前5行和前5列：

```
> dim(rf$proximity)
[1] 178 178

> rf$proximity[1:5, 1:5]
          1         2         3          4          5
1 1.0000000 0.2593985 0.2953586 0.36013986 0.17054264
2 0.2593985 1.0000000 0.1307420 0.16438356 0.11029412
3 0.2953586 0.1307420 1.0000000 0.29692833 0.23735409
4 0.3601399 0.1643836 0.2969283 1.00000000 0.08076923
5 0.1705426 0.1102941 0.2373541 0.08076923 1.00000000
```

对这些值的一种理解方式是，将它们看作两个观测出现在同一个终端节点的次数百分比！检查变量重要性后可知，完全能够丢弃“酒精含量”这个被转换过的输入特征。为简单起见，我们依然保留它：

```
> importance(rf)
             MeanDecreaseGini
Alcohol      0.5614071
MalicAcid    6.8422540
Ash          6.4693717
Alk_ash      5.9103567
magnesium    5.9426505
T_phenols    6.2928709
Flavanoids   6.2902370
Non_flav     5.7312940
Proantho     6.2657613
C_Intensity  6.5375605
Hue          6.3297808
OD280_315    6.4894731
Proline      6.6105274
```

下面只剩下创建相异度矩阵了，需要对邻近度的值转换如下（(1 – 邻近度)的平方根）：

```
> dissMat <- sqrt(1 - rf$proximity)

> dissMat[1:2, 1:2]
          1         2
1 0.0000000 0.8605821
2 0.8605821 0.0000000
```

这样就得到了聚类算法的输入特征，所以可以运行PAM聚类算法了，和以前一样：

```
> set.seed(123)

> pamRF <- pam(dissMat, k = 3)

> table(pamRF$clustering)

 1  2  3
62 68 48
> table(pamRF$clustering, wine$Class)

   1  2  3
 1 57  5  0
 2  2 64  2
 3  0  2 46
```

上面的结果和使用其他技术得到的结果很相似。你能通过对随机森林进行调优来改善这个结果吗？

如果你的聚类数据是一团乱麻，那么可以考虑随机森林方法。

8.8　小结

我们在本章开始研究无监督学习技术，重点讨论了聚类分析。它既可以进行数据降维，也可以理解观测数据。

我们介绍了4种方法：传统的层次聚类、K均值聚类算法、PAM方法，还有可以处理两种不同类型输入数据（果瓦系数和随机森林）的方法。在来自3种不同品种等级的意大利葡萄酒数据集上，我们应用了这4种方法来寻找潜在的数据结构，并对结果进行了研究。

下一章继续研究无监督学习方法，但不再满足于寻找观测中的潜在结构，重点在于通过找到变量的潜在结构来构造新的特征，并将新的特征用于监督式学习。

第9章 主成分分析

“有的人追着球跑，我则守候在球的必经之路上。”

——韦恩·格雷茨基

我们在本章依旧致力于讲解无监督学习技术。前一章介绍了聚类分析，它可以将相似的观测归成一类。在这一章，我们将研究**主成分分析**（PCA），它可以对相关变量进行归类，从而降低数据维度，提高对数据的理解。此后，我们会将主成分用于监督式学习。

在很多数据集中，特别是社会科学领域的数据集中，你会发现很多彼此相关的变量。此外，高维度也会带来问题，这就是所谓的**维数灾难**，因为模型估计所需的样本数量是随着输入特征的数量指数增长的。这种数据集中会出现某些变量冗余，因为这些变量最后起的作用与其他变量基本是重复的。比如收入水平与贫穷程度，或者抑郁度与焦虑度。那么我们的目标就是，通过PCA从原始变量集合中找出一个更小的，但是能保留原来大部分信息的变量集合。这样可以简化数据集，并经常能够发现数据背后隐藏的知识。这些新的变量（主成分）彼此高度不相关，除了可以用于监督式学习之外，还经常用于数据可视化。

我使用PCA进行分析已经有十多年了，期间的一个感受就是，PCA虽然被广泛使用，但真正理解它的人却很少，特别是在那些不做分析而只是享受成果的人当中。PCA可以直观地理解为，使用一些相关的变量构造一个新的变量。但是，由于对术语的错误理解，人们并不经常充分使用这项技术，其中的数学概念也使非专业人员无所适从。本章的目的就是说清楚PCA的概念和使用方法，包括以下内容：

- 为PCA准备好一个数据集
- 执行PCA
- 选择主成分
- 使用主成分建立一个预测模型
- 使用预测模型进行样本外预测

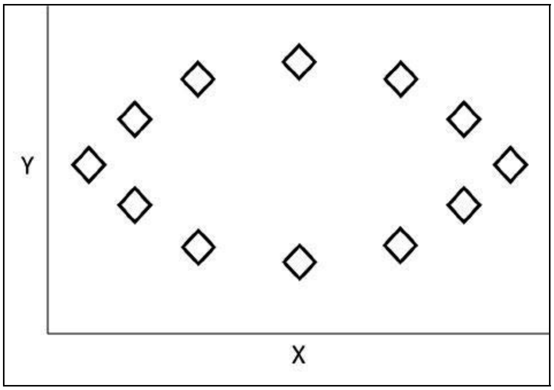
9.1　主成分简介

主成分分析就是寻找主成分的过程。那么，主成分到底是什么？

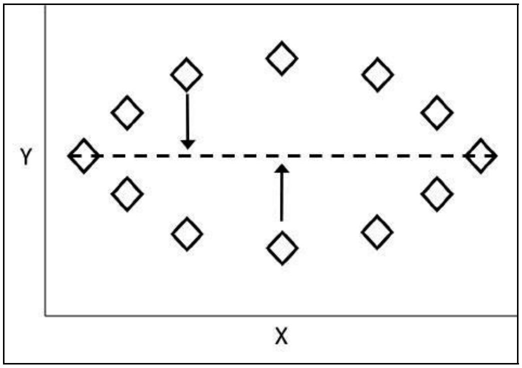
可以认为成分就是特征的规范化线性组合（James，2012）。在一个数据集中，第一主成分就是能够最大程度解释数据中的方差的特征线性组合。第二主成分是另一种特征线性组合，它在方向与第一主成分垂直这个限制条件下，最大程度解释数据中的方差。其后的每一个主成分（可以构造与变量数相等数目的主成分）都遵循同样的规则。

要注意两件事。PCA定义提到了**线性组合**，这是一个关键假设。如果你试图在一个变量之间基本不相关的数据集上使用PCA，很可能会得到一个毫无意义的分析结果。另外一个关键假设是，变量的均值和方差是充分统计量。也就是说，数据应该服从正态分布，这样协方差矩阵即可充分描述数据集。换言之，数据要满足**多元正态分布**。PCA对于非正态分布的数据具有相当强的鲁棒性，甚至可以和二值变量一起使用，所以结果具有很好的解释性。

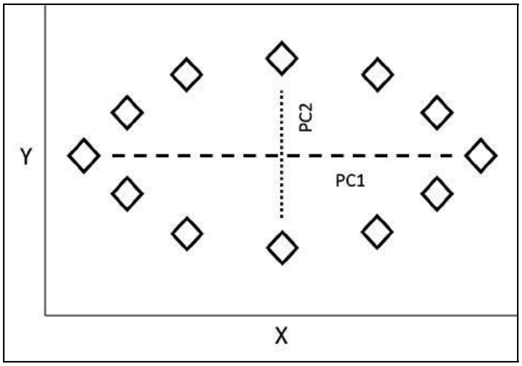
那么，这里说的“方向”是什么？如何确定特征的线性组合呢？理解PCA的最好方式就是可视化。假设有一个小数据集，其中有两个变量，我们可以画出该数据集的分布图。PCA对量度是敏感的，所以数据已经被标准化为均值为0、方差为1。你可以在下面的图中看到，每个菱形代表一个观测，数据正好组成一个椭圆形状。



从图中可知，数据在X轴方向具有最大的方差，所以我们可以画一条水平短划线来表示**第一主成分**，如下图所示。这个主成分是两个变量的线性组合，或表示为$PC_1 = \alpha_{11}X_1 + \alpha_{12}X_2$，这里的系数权重是这个主成分中的变量载荷。该主成分建立了一个基准方向，在这个方向上，数据差异最大。上面的等式中有个限制——系数的平方和为1，这是为了防止随意选择过高的值。主成分的另一种理解方式是，短划线使数据点到它本身的距离最小。随意选出两个点，用箭头表示它们到直线的距离，如下页图所示。



可以用同样的方式得出**第二主成分**，只是它应该与第一主成分不相关。也就是说，它的方向与第一主成分是垂直的。下图显示第二主成分，用点线表示。



通过为每个变量计算出主成分载荷，算法可以为我们提供主成分得分。主成分得分是对于每个主成分和每个观测计算的。对于PC_1和第一个观测，主成分得分公式为$Z_{11} = \alpha_{11} \times (X_{11} - X_1$的平均数$) + \alpha_{12} \times (X_{12} - X_2$的平均数$)$。对于$PC_2$和第一个观测，公式为$Z_{12} = \alpha_{21} \times (X_{11} - X_1$的平均数$) + \alpha_{22} \times (X_{12} - X_2$的平均数$)$。这些主成分得分就构成了新的特征空间，你可以使用它们进行各种分析。

9

我们前面说过，算法可以构造与变量数相同的主成分，这样可以解释100%的方差。那么应该如何精简主成分，来达到降低数据维度这一首要初始目标呢？可以使用一些启发式方法，在下面的建模过程中，我们介绍一种专业但是通用的方法，在**特征值**大于1的情况下选择主成分。估计特征值和**特征向量**所需的线性代数知识超过了本书范围，尽管如此，讨论它们的概念和在PCA中的应用还是很重要的。

> 最优线性权重是通过线性代数运算得到特征向量而求出的，它们是最优解，因为没有其他可能的权重组合可以比它们更好地解释方差。主成分的特征值是它在整个数据集中能够解释的方差的数量。

回忆一下第一主成分的计算公式是$PC_1 = \alpha_{11}X_1 + \alpha_{12}X_2$。

因为第一特征值可以解释最大数量的方差，它就有最大的特征值；第二主成分有第二大的特征值，依此类推。所以，特征值大于1就表示这个主成分解释的方差比任何一个原始变量都要大。如果通过标准化操作将特征值的总和变为1，就能够得到每个主成分解释的方差的比例。这也有助于确定一个适当的分界点。

特征值原则并不严格、明确，它必须和你的数据分析知识以及实际业务问题结合起来。如果你已经选定了主成分的数量，就可以对主成分进行旋转处理，以简化对它们的解释。

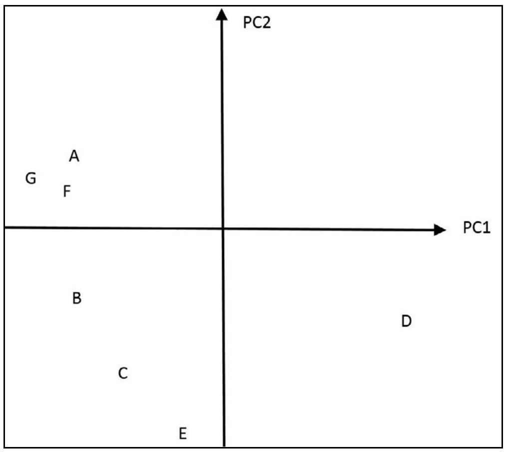
主成分旋转

应不应该做旋转？前面提到过，旋转可以修改每个变量的载荷，这样有助于对主成分的解释。旋转后的成分能够解释的方差总量是不变的，但是每个成分对于能够解释的方差总量的贡献会改变。在旋转过程中，你会发现载荷的值或者更远离0，或者更接近0，这在理论上可以帮助我们识别那些对主成分起重要作用的变量。这是一种将变量和唯一一个主成分联系起来的尝试。请记住，PCA是一种无监督学习，所以你是在努力去理解数据，而不是在验证某种假设。总之，旋转有助于你的这种努力。

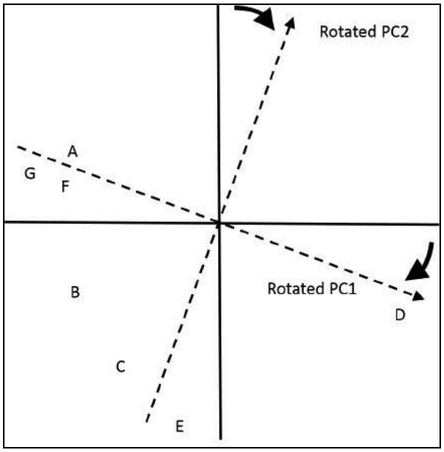
最常用的主成分旋转方法被称为**方差最大法**。虽然还有其他方法，比如**四次方最大法**和**等量最大法**，但我们主要讨论方差最大旋转。根据我的经验，其他方法从来没有提供过比方差最大法更好的解。当然，你可以通过反复实验来决定使用哪种方法。

在方差最大法中，我们要使平方后的载荷的总方差最大。方差最大化过程会旋转特征空间的轴和坐标，但不改变数据点的位置。

演示旋转过程的最好方式是通过一个简单的图示。假设一个数据集中有A~G这7个变量，有两个主成分。画出这些数据，可以得到下图。



为了便于讨论，假设变量A在PC1上的载荷是–0.4，在PC2上的载荷是0.1；变量D在PC1上的载荷是0.4，在PC2上的载荷是–0.3；对于E点，载荷分别是–0.05和–0.7。请注意，载荷的符号与主成分的方向是一致的。运行方差最大化过程，旋转后的主成分如下图所示。



下面是旋转后的PC1和PC2上的新的载荷。

- 变量A：–0.5和0.02
- 变量D：0.5和–0.3
- 变量E：0.15和–0.75

载荷发生了变化，但数据点没有变。通过这个简单的图示，我们还不能说已经简化了对主成分的解释，但可以帮助你理解主成分旋转过程中发生了什么。

9.2 业务理解

在这个案例中，我们开始探究体育世界，具体地说，是美国国家冰球大联盟。已经有人在棒球（参见畅销书和电影《点球成金》）和橄榄球上进行了很多研究，二者都是地道的美国式运动，并且被全世界人民所喜爱。但在我看来，没有比冰球更激动人心的运动了，可能这就是生长于北达科他州冰冻荒原才能获得的特有恩赐吧。不管怎么说，我都可以从这个分析开始我的冰球淘金之旅。

在这个分析中，我们要研究30支大联盟球队的统计数据，这个数据集中的数据是我从www.nhl.com和www.puckalytics.com这两个网站上整理的。我们的目标是建立一个模型来预测一只队伍的总积分，通过PCA建立一个输入特征空间，目的是揭示哪些因素能够造就一支顶级职业球队。首先从2015~2016赛季的数据中学习出一个模型，在这个赛季，匹兹堡企鹅队最终加冕冠军，然后，使用当前赛季至2017年2月15日为止的结果来检验这个模型的性能。数据文件的名称是nhlTrain.csv和nhltest.csv，地址为https://github.com/datameister66/data/。

NHL基于一个计分系统对球队进行排名，所以我们的预测结果就是球队每场比赛的得分。清楚NHL如何给球队奖励积分是非常重要的。与橄榄球及棒球不同，它们只计算胜场数和负场数，职业冰球对每场比赛使用下面的积分规则：

- 胜者得2分，不论是在常规时间、加时还是加时后的点球大战中获胜；
- 常规时间的负者得0分；
- 加时赛或点球大战的负者得1分，这就是所谓**负者分**。

NHL从2005年开始使用这种积分系统，这种系统并非没有争议，但它确实没有减少这项运动中优雅而又得体的力量与激情。

数据理解与数据准备

为了下载数据和进行后面的分析，先加载必需的程序包。在加载之前，请确保你已经安装完毕：

```
> library(ggplot2) #support scatterplot

> library(psych) #PCA package
```

假设你已经将两个.csv文件保存到工作目录，那么可以使用`read.csv()`函数读取训练数据：

```
> train <- read.csv("NHLtrain.csv")
```

使用结构函数`str()`检查数据。为了节省篇幅，我只列出函数输出的开始几行：

```
> str(train)
'data.frame': 30 obs. of 15 variables:
$ Team : Factor w/ 30 levels "Anaheim","Arizona",..: 1 2 3 4 5 6 7
  8 9 10 ...
$ ppg : num 1.26 0.95 1.13 0.99 0.94 1.05 1.26 1 0.93 1.33 ...
$ Goals_For : num 2.62 2.54 2.88 2.43 2.79 2.39 2.85 2.59 2.6 3.23
  ...
$ Goals_Against: num 2.29 2.98 2.78 2.62 3.13 2.7 2.52 2.93 3.02
  2.78 ...
```

下面需要查看变量名：

```
    > names(train)
[1] "Team" "ppg" "Goals_For" "Goals_Against" "Shots_For"
    [6] "Shots_Against" "PP_perc" "PK_perc" "CF60_pp" "CA60_sh"
    [11] "OZFOperc_pp" "Give" "Take" "hits" "blks"
```

以下是其各自的含义。

- `Team`：球队所在城市。
- `ppg`：平均每场得分，得分规则如前所述。
- `Goals_For`：平均每场进球数。

- Goals_Against：平均每场失球数。
- Shots_For：平均每场射中球门次数。
- Shots_Against：平均每场被射中球门次数。
- PP_perc：球队获得以多打少机会时的进球百分比。
- PK_perc：对方获得以多打少机会时，球队力保球门不失的时间百分比。
- CF60_pp：球队在每60分钟以多打少时间内获得的Corsi分值；Corsi分值是射门次数总和，包括射中球门次数（Shots_For）、射偏次数和被对方封堵的次数。
- CA60_sh：对方以多打少时，即本方人数劣势时，对方每60分钟获得的Corsi分值。
- OZFOperc_pp：球队以多打少时，在进攻区域发生的争球次数百分比。
- Give：平均每场丢球次数。
- Take：平均每场抢断次数。
- hits：平均每场身体冲撞次数。
- blks：平均每场封堵对方射门次数。

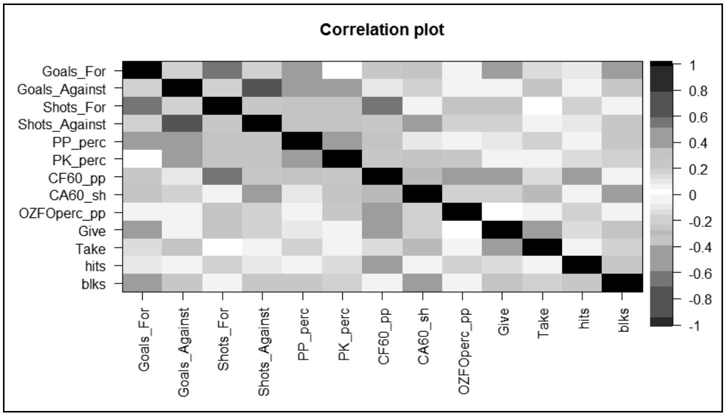
我们需要对数据进行标准化，使数据的均值为0，标准差为1。完成标准化后，使用psych包提供的cor.plot()函数，创建一个输入特征的相关性统计图：

```
> train.scale <- scale(train[, -1:-2])

> nhl.cor <- cor(train.scale)

> cor.plot(nhl.cor)
```

上述命令输出如下。



可以看出一些有意思的事情。我们发现Shots_For与Goals_For相关，反之，Shots_Against与Goals_Against也相关。PP_perc及PK_perc与Goals_Against之间存在某种负相关。

由此可知，这个数据集非常适合提取主成分。

请注意，这些特征（或变量）是我根据自己的兴趣选择的，你还可以选择各种不同的统计量，看看能否提高预测能力。

9.3　模型构建与模型评价

对于模型构建过程，我们按照以下几个步骤进行：

(1) 抽取主成分并决定保留的数量；
(2) 对留下的主成分进行旋转；
(3) 对旋转后的解决方案进行解释；
(4) 生成各个因子的得分；
(5) 使用得分作为输入变量进行回归分析，并使用测试数据评价模型效果。

R中有许多方法和程序包可以进行主成分分析，其中最常用的是R基础包中的`prcomp()`和`princomp()`函数。但是在我看来，psych包最灵活，它有最好的选项。

9.3.1　主成分抽取

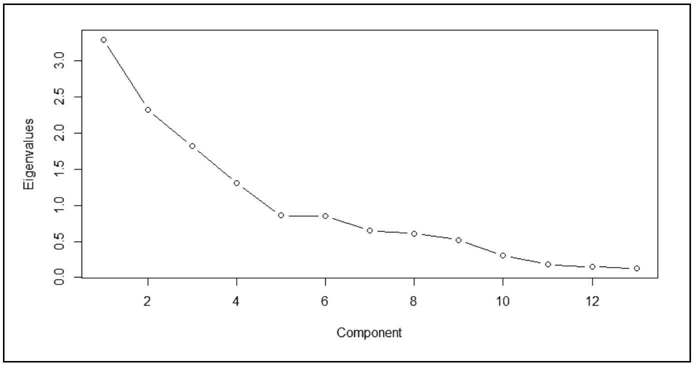
通过psych包抽取主成分要使用`principal()`函数，这个函数的语法中要包括数据和是否要进行主成分旋转：

```
> pca <- principal(train.scale, rotate="none")
```

可以调用函数生成的pca对象检查各个成分，但我们的主要目的是确定要保留的成分的数量。为此，使用碎石图即可。碎石图可以帮助你评估能解释大部分数据方差的主成分，它用X轴表示主成分的数量，用Y轴表示相应的特征值：

```
> plot(pca$values, type="b", ylab="Eigenvalues", xlab="Component")
```

上述命令输出如下。



需要在碎石图中找出使变化率降低的那个点，也就是我们常说的统计图中的“肘点”或弯曲点。在统计图中，肘点表示在这个点上新增加一个主成分时，对方差的解释增加得并不太多。换句话说，这个点就是曲线由陡变平的转折点。从这个图中可以看出，5个主成分是很令人信服的。

我从多年经验中总结出的另外一条原则是，你应该解释总数70%左右的方差。这意味着你选择的主成分解释的方差累加起来，应该能够解释70%所有成分解释的方差。

9.3.2 正交旋转与解释

我们在前面提到过，旋转背后的意义是使变量在某个主成分上的载荷最大化，这样可以减少（或消灭）主成分之间的相关性，有助于对主成分的解释。进行正交旋转的方法称为“方差最大法”。还有其他非正交旋转方法，这种方法允许主成分（因子）之间存在相关性。如何在实际工作中选择旋转方法需要参考相关文献，这已经超出了本书范围。你可以使用这个数据集做一些实验，但我认为，无法做出明确选择时，应该选择正交旋转作为主成分分析的起点。

要想进行正交旋转，依然要使用`principal()`函数，语法要稍作修改。我们设定使用5个主成分，并需要进行正交旋转。如下所示：

```
> pca.rotate <- principal(train.scale, nfactors = 5, rotate =
  "varimax")

> pca.rotate
Principal Components Analysis
Call: principal(r = train.scale, nfactors = 5, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation
  matrix

                RC1   RC2    RC5   RC3   RC4   h2   u2  com
Goals_For     -0.21  0.82   0.21  0.05 -0.11 0.78 0.22  1.3
Goals_Against  0.88 -0.02  -0.05  0.21  0.00 0.82 0.18  1.1
Shots_For     -0.22  0.43   0.76 -0.02 -0.10 0.81 0.19  1.8
Shots_Against  0.73 -0.02  -0.20 -0.29  0.20 0.70 0.30  1.7
PP_perc       -0.73  0.46  -0.04 -0.15  0.04 0.77 0.23  1.8
PK_perc       -0.73 -0.21   0.22 -0.03  0.10 0.64 0.36  1.4
CF60_pp       -0.20  0.12   0.71  0.24  0.29 0.69 0.31  1.9
CA60_sh        0.35  0.66  -0.25 -0.48 -0.03 0.85 0.15  2.8
OZFOperc_pp   -0.02 -0.18   0.70 -0.01  0.11 0.53 0.47  1.2
Give          -0.02  0.58   0.17  0.52  0.10 0.65 0.35  2.2
Take           0.16  0.02   0.01  0.90 -0.05 0.83 0.17  1.1
hits          -0.02 -0.01   0.27 -0.06  0.87 0.83 0.17  1.2
blks           0.19  0.63  -0.18  0.14  0.47 0.70 0.30  2.4

                       RC1  RC2   RC5  RC3  RC4
SS loadings           2.69 2.33  1.89 1.55 1.16
Proportion Var        0.21 0.18  0.15 0.12 0.09
Cumulative Var        0.21 0.39  0.53 0.65 0.74
Proportion Explained  0.28 0.24  0.20 0.16 0.12
Cumulative Proportion 0.28 0.52  0.72 0.88 1.00
```

输出中有两个部分比较重要，第一部分就是5个主成分中每个主成分的变量载荷，分别标注为RC1至RC5。我们看到，对于第一个主成分，变量Goals_Against和Shots_Against具有非常高的正载荷，而PP_perc和PK_perc具有高的负载荷。对于第二个主成分，具有高载荷的是Goals_For。第五个主成分在Shots_For、ff和OZFOperc_pp上具有高载荷。第三个主成分看上去只与变量take有关系，第四个主成分则只与hits有关。下面看一下第二个重要部分，就是以平方和SS loading开始的表格。SS loading中的值是每个主成分的特征值。如果对特征值进行标准化，就可以得到Proportion Explained行。你应该已经猜到，这一行表示的是每个主成分解释的方差的比例。可以看到，对于旋转后的5个主成分能够解释的所有方差，第一个主成分可以解释其中的28%。回忆一下前面提到的经验原则，你选择的主成分应该至少解释大约70%的全部方差。查看Cumulative Var行可以知道，这5个旋转后的主成分可以解释74%的全部方差。所以我们可以充满信心地认为，已经找到了合适数量的主成分，可以进行下一步的建模工作了。

9.3.3　根据主成分建立因子得分

现在检查旋转后的主成分载荷，并将其作为每个球队的因子得分。这些得分说明了每个观测（在我们的案例中是NHL球队）与旋转后的主成分的相关程度。查看得分并将其保存到数据框，因为要使用它们进行回归分析：

```
> pca.scores <- data.frame(pca.rotate$scores)

> head(pca.scores)

          RC1          RC2        RC5        RC3        RC4
1 -2.21526408  0.002821488  0.3161588 -0.1572320  1.5278033
2  0.88147630 -0.569239044 -1.2361419 -0.2703150 -0.0113224
3  0.10321189  0.481754024  1.8135052 -0.1606672  0.7346531
4 -0.06630166 -0.630676083 -0.2121434 -1.3086231  0.1541255
5  1.49662977  1.156905747 -0.3222194  0.9647145 -0.6564827
6 -0.48902169 -2.119952370  1.0456190  2.7375097 -1.3735777
```

得到每个球队在每个因子上的得分，这些得分的计算非常简单，每个观测的变量值乘以载荷然后相加即可。现在可以将响应变量（ppg）作为一列加入数据：

```
> pca.scores$ppg <- train$ppg
```

做完这项工作之后，即可开始建模预测。

9.3.4　回归分析

要完成这部分内容，只需重复第2章中的步骤和代码。如果你没有学习第2章，那么请回过头去学习一下，这样才能看懂下面的输出结果。

通过lm()函数建立线性模型，使用所有因子作为输入，然后查看结果摘要：

```
> nhl.lm <- lm(ppg ~ ., data = pca.scores)

> summary(nhl.lm)

Call:
lm(formula = ppg ~ ., data = pca.scores)

Residuals:
      Min        1Q    Median       3Q      Max
-0.163274 -0.048189 0.003718 0.038723 0.165905

Coefficients:
      Estimate           Std. Error t value Pr(>|t|)
(Intercept)     1.111333   0.015752  70.551  < 2e-16  ***
RC1            -0.112201   0.016022  -7.003  3.06e-07 ***
RC2             0.070991   0.016022   4.431  0.000177 ***
RC5             0.022945   0.016022   1.432  0.164996
RC3            -0.017782   0.016022  -1.110  0.278044
RC4            -0.005314   0.016022  -0.332  0.743003
---
Signif. codes: 0  '***'  0.001  '**'  0.01  '*'  0.05  '.'  0.1  ' '  1

Residual standard error: 0.08628 on 24 degrees of freedom
Multiple R-squared: 0.7502, Adjusted R-squared: 0.6981
F-statistic: 14.41 on 5 and 24 DF, p-value: 1.446e-06
```

好消息是，我们的整体模型在统计上是高度显著的，p值为1.446e－06，修正R方几乎是70%。坏消息是，有3个主成分是不显著的。可以简单处理，选择将其保留在模型中。但是我们先看看如果把它们排除出模型，只保留RC1和RC2，会发生什么：

```
> nhl.lm2 <- lm(ppg ~ RC1 + RC2, data = pca.scores)

> summary(nhl.lm2)

Call:
lm(formula = ppg ~ RC1 + RC2, data = pca.scores)

Residuals:
     Min       1Q   Median      3Q     Max
-0.18914 -0.04430 0.01438 0.05645 0.16469

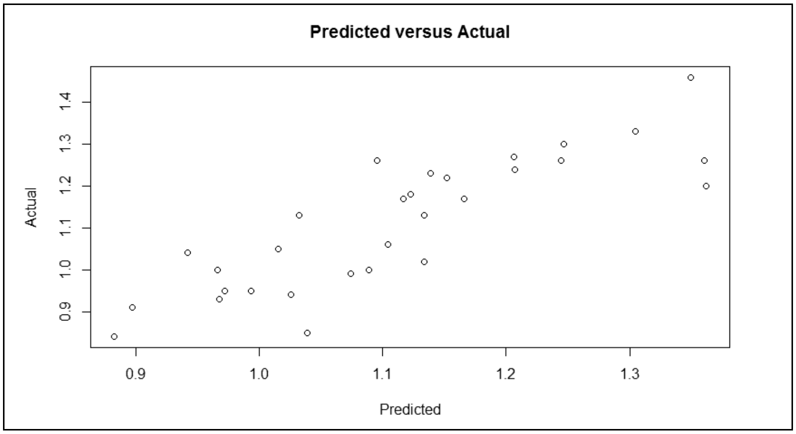
Coefficients:
           Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.11133    0.01587  70.043  < 2e-16 ***
RC1        -0.11220    0.01614  -6.953  1.8e-07 ***
RC2         0.07099    0.01614   4.399  0.000153 ***
---
Signif. codes: 0  '***'  0.001  '**'  0.01  '*'  0.05  '.'  0.1  ' '  1

Residual standard error: 0.0869 on 27 degrees of freedom
Multiple R-squared: 0.7149, Adjusted R-squared: 0.6937
F-statistic: 33.85 on 2 and 27 DF, p-value: 4.397e-08
```

模型还是能得到一个几乎一样的修正R方（69.37%）值，因子的系数在统计上也是显著的。诊断测试的细节就不介绍了，代之以统计图来更加深入地进行分析。可以通过R基础包中的图形功能生成一张散点图，查看预测值和实际值（所有球队的积分）之间的关系。如下所示：

```
> plot(nhl.lm2$fitted.values, train$ppg,
  main="Predicted versus Actual",
  xlab="Predicted",ylab="Actual")
```

上述命令输出如下。



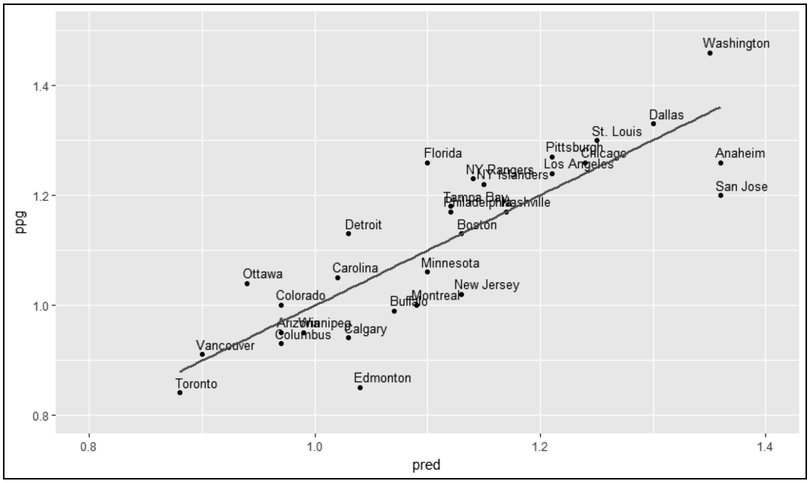
这张图证实，我们的模型在使用两个因子预测球队的比赛结果方面表现得非常好，它也凸显了主成分和球队积分之间存在强烈的线性相关性。再进一步，使用ggplot2包生成一张带有球队名字的散点图。唯一的问题是，这个函数功能非常强大，里面的设置非常多。有很多在线资源可以为你指点迷津，但是下面的代码可以帮助你快速实现。首先生成基准图，并将它赋给一个名为p的对象，然后添加各种绘图功能。

```
> train$pred <- round(nhl.lm2$fitted.values, digits = 2)

> p <- ggplot(train, aes(x = pred,
  y = ppg,
  label = Team))

> p + geom_point() +
  geom_text(size = 3.5, hjust = 0.1, vjust = -0.5, angle = 0) +
  xlim(0.8, 1.4) + ylim(0.8, 1.5) +
  stat_smooth(method = "lm", se = FALSE)
```

上述命令输出如下页图。



建立对象p的代码非常简单，只需指定数据框，在`aes()`中设定X轴和Y轴的变量，并指定使用哪个变量作为标签即可。下面将基准图美化一下，先加上数据点。在代码中使用+操作符，可以在图中加上任何需要的内容。如下所示：

```
> p + geom_point() +
```

然后设置球队标签的显示方式。需要多试几次，以确定合适的字体大小和位置：

```
geom_text() +
```

在这之后，可以设定X轴和Y轴的界限，否则统计图中就不会出现落到界限之外的那些观测点。如下所示：

```
xlim() + ylim() +
```

最后，添加一条不带标准差的最佳拟合线：

```
stat_smooth(method = "lm", se = FALSE)
```

我认为可以这样解释这张图：位于斜线下方的球队发挥欠佳，位于斜线上方的球队则超过预期。

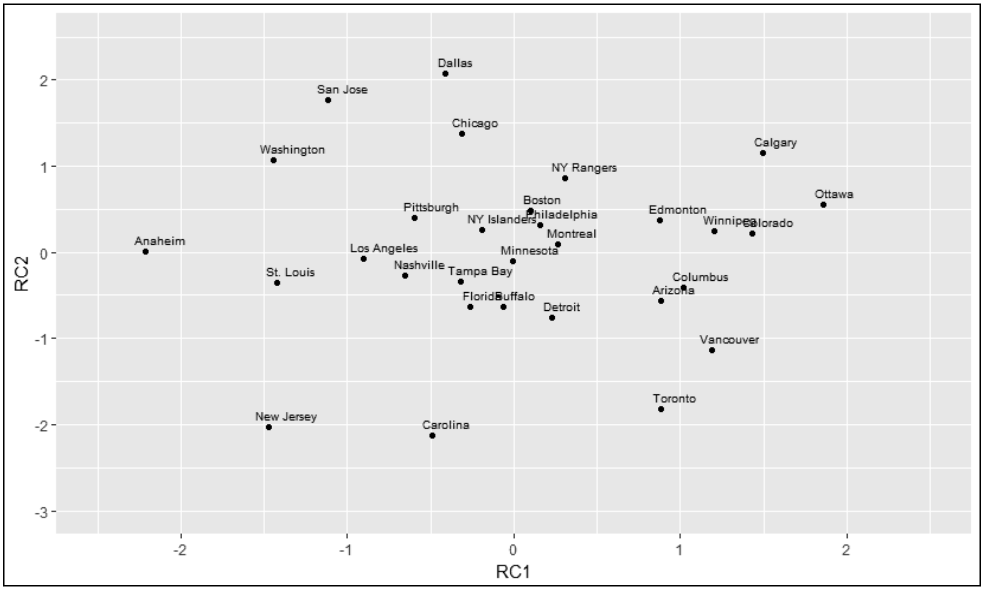
另一项分析内容是绘制出球队及其因子得分之间的关系，这样的图称为**双标图**。依然使用`ggplot()`来帮助分析。以前面的示例代码为基础，修改设置并查看结果：

```
> pca.scores$Team <- train$Team

> p2 <- ggplot(pca.scores, aes(x = RC1, y = RC2, label = Team))

> p2 + geom_point() +
  geom_text(size = 2.75, hjust = .2, vjust = -0.75, angle = 0) +
  xlim(-2.5, 2.5) + ylim(-3.0, 2.5)
```

上述命令输出如下。



可以看出，X轴是球队在RC1上的得分，Y轴则是RC2上的得分。看一下“阿纳海姆小鸭队”，它在RC1上的分数最低，在RC2上的分数位于中游。考虑一下，这意味着什么。在RC1上，以多打少进球（`PP_perc`）和以少打多失球（`PK_perc`）具有负载荷，平均每场失球数（`Goals_Against`）具有正载荷，这说明这支球队的防守组织得非常好，并在处于人数劣势时表现得很好。顺便说一句，“匹兹堡企鹅队”最终获得了这赛季的斯坦利杯。他们的分数很实在，但也没什么出奇之处。请注意，这支球队在赛季初有个噩梦般的开始，并解雇了原来的教练。如果对他们上半赛季和下半赛季的表现做一番分析和比较，那会非常有意思。

像之前做过的那样，你还可以评价模型误差。下面看看**均方根误差**。

```
> sqrt(mean(nhl.lm2$residuals^2))
[1] 0.08244449
```

这些工作完成之后，看看这个模型在样本外数据上的效果。需要加载测试数据，通过主成分预测球队得分，然后基于线性模型做出预测。psych包中的`predict()`函数会自动对测试数据进行标准化：

```
> test <- read.csv("NHLtest.csv")

> test.scores <- data.frame(predict(pca.rotate, test[, c(-1:-2)]))

> test.scores$pred <- predict(nhl.lm2, test.scores)
```

我觉得应该和前面一样，将结果绘制出来并标上球队名称。先将所有信息放在一个数据框中：

```
> test.scores$ppg <- test$ppg

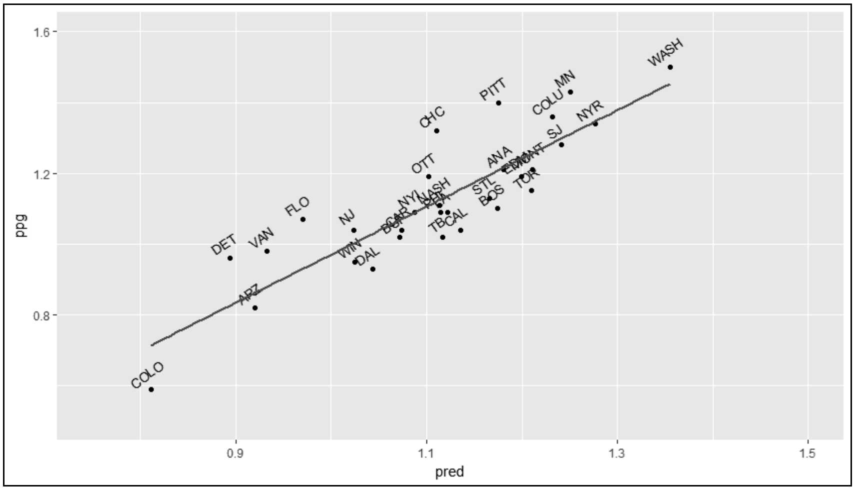
> test.scores$Team <- test$Team
```

然后让ggplot()大显身手：

```
> p <- ggplot(test.scores, aes(x = pred,
                               y = ppg,
                               label = Team))

> p + geom_point() +
  geom_text(size=3.5, hjust=0.4, vjust = -0.9, angle = 35) +
  xlim(0.75, 1.5) + ylim(0.5, 1.6) +
  stat_smooth(method="lm", se=FALSE)
```

上述命令输出如下。



我对球队名称进行了缩写，使这幅图更加清晰易读。在平均每场得分上，“华盛顿首都”队一马当先，“科罗拉多雪崩”队则叨陪末座。实际上，在我采集这份数据的时候，“科罗拉多雪崩”队已经连续输掉了5场比赛，直到他们通过加时赛击败“卡罗莱纳飓风”队，才止住了连败势头。

最后，再检查一下RMSE。

```
> resid <- test.scores$ppg - test.scores$pred

> sqrt(mean(resid^2))
[1] 0.1011561
```

与样本内误差0.08比起来，0.1的样本外误差并不坏。我认为，可以宣布这个模型是有效的。但还有很多球队统计数据可以添加到模型中，以提高预测能力和减小误差。我会一直致力于完善这个模型，希望你也同样如此。

9.4 小结

本章再次讨论了无监督学习技术,研究了主成分分析,介绍其概念并对这种技术进行了应用。我们研究了如何使用PCA降低数据的维度和提高对数据的理解,特别当数据具有很多高度相关的变量时。然后,我们在一个来自美国国家冰球联盟的真实数据集上应用了这种技术,并使用从数据集中抽取的主成分进行了回归分析,以预测球队的得分。此外,我们还介绍了对数据和主成分进行可视化的方法。

作为一种无监督学习技术,主成分分析需要一定的判断能力以及反复实验,才能得到被商业伙伴所接受的最优解。尽管如此,主成分分析依然是一种可以挖掘潜在知识并支持监督式学习的强大技术。

下一章将介绍如何使用无监督学习技术进行购物篮分析和实现推荐引擎,PCA会在其中发挥重要作用。

第 10 章

购物篮分析、推荐引擎与序列分析

10

“要想使你的业务量翻一番，就应该使你的顾客转化率翻一番，这比使客流量翻一番容易多了。”

——杰夫·艾斯伯格，BuyerLegend.com首席执行官

“在Whole Foods超市中，我看不到人们的脸上有笑容。”

——沃伦·巴菲特

如果有人没有感受到本章讨论的技术所带来的影响，那他一定是生活在了月球背面。如果你访问过www.amazon.com，或在www.netflix.com上看过电影，或者访问过任何一个零售网站，肯定会经常看到一些名词，比如“相关商品”“因为你看过……”“购买了x的顾客也购买了y”“向您推荐”等，这些名词随处可见。掌握了你在系统中的历史信息和近期记录后，零售商可以使用本章将要讨论的算法来增加顾客的购物次数和购买总量。

这些技术可以分为两类：关联规则和推荐引擎。关联规则分析通常称为购物篮分析，因为我们试图发现人们会同时购买哪些商品。推荐引擎的目的是基于顾客对以前浏览过或购买过的商品的评价，向他们推荐可能感兴趣的其他商品。

另外一种应用于商业的技术是，对你在购买商品或使用服务中的一系列行为进行理解和分析，这种技术称为序列分析。该方法的一个常见实例就是，理解和分析顾客如何在各种网页和链接之间点来点去。

在下面的案例中，我们将研究如何使用R实现这样的算法。具体的实现细节不做介绍，因为这超出了本书范围。我们从一个对零售商店购物习惯的购物篮分析开始，然后深入研究如何基于网站评论建立推荐引擎，最后分析用户在网页之间的序列行为。

10.1　购物篮分析简介

购物篮分析是一项数据挖掘技术，它的目标是找到最优的商品与服务组合，使市场营销人员以此为依据提供推荐意见，优化商品摆放，制定营销方案，最终提高交叉销售。简而言之，它的理念就是识别出哪些商品放在一起更好，并从中获益。

你可以把这种分析结果视为某种`if...then`语句。如果一个顾客买了一张飞机票，那么他就有46%的可能性在旅馆订一个房间；如果他确实在旅馆订了一个房间，那么他就有33%的可能性租一辆轿车。

然而，购物篮分析不仅能用于销售和营销，还可以用于欺诈检测和医疗保健。举例来说，如果一个患者正在按A方案进行治疗，那么他就有26%的可能性会表现出症状X。进入详细讨论之前，先看几个将会出现在案例中的术语。

- **项集**：数据集中一个或多个项目的集合。
- **支持度**：包含某个项集的事务在整个数据中的比例。
- **置信度**：如果某人购买了x（或做了x），那么他就会购买y（或做y）的条件概率；x被称为**先导**或**左侧项**，y被称为**后继**或**右侧项**。
- **提升度**：它是一个比例，x发生的同时发生y的支持度是分子，分母是x和y在相互独立的情况下同时发生的概率。它等于置信度/(x的概率 × y的概率)。举例来说，假设x和y同时发生的概率是10%，x发生的概率是20%，y发生的概率是30%，那么提升度就是10%/(20% × 30%)，等于1.667%。

在R中，可以用来进行购物篮分析的软件包是**arules：挖掘关联规则和频繁集**。这个R包提供了两种不同的关联规则发现方法。为什么对一种技术提供两种方法呢？答案很简单，如果数据集规模很大，那么检查所有可能的商品组合将非常耗费计算成本。这个R包支持Apriori算法和ECLAT算法。还有其他算法可以进行购物篮分析，但Apriori最常用，所以我们重点讨论这种算法。

Apriori算法的主要原理就是，如果一个项集是频繁集，那么它的所有子集必然也是频繁集。最小频率（支持度）是由分析者在执行算法之前确定的，确定了最小支持度之后，算法将按照下面的步骤运行：

- 使$k = 1$（项数）；
- 按照项数生成大于等于最小支持度的项集；
- 按照$k + (1 \cdots n)$进行迭代，删除那些不频繁（小于最小支持度）的项集；
- 没有新的频繁集产生时，停止迭代。

当你得到排好序的频繁集列表之后，可以通过检查置信度和提升度来继续分析，目的是发现有价值的关联规则。

10.2 业务理解

在我们的案例中，重点在于如何为一个杂货店发现关联规则。数据集来自arules包，名为Groceries。这个数据集包含了一个真实杂货店的30天交易信息，共有9835条购买记录。所有售出商品被分成169类，比如面包、葡萄酒、肉类等。

假设我们是一家刚刚起步的小啤酒厂，正在努力提高在这个杂货店的销售量。我们试图弄清楚潜在顾客在购买啤酒的同时还会购买什么商品。这种知识可以帮助我们找出商品在商店中的正确摆放方式，或支持交叉销售活动。

10.3 数据理解和数据准备

为了进行分析，只需加载两个R包，以及Groceries数据集：

```
    > library(arules)

> library(arulesViz)

> data(Groceries)

> head(Groceries)

transactions in sparse format with
 9835 transactions (rows) and
 169 items (columns)

    > str(Groceries)
    Formal class 'transactions' [package "arules"] with 3 slots
      ..@ data :Formal class 'ngCMatrix' [package "Matrix"] with 5
        slots
      .. .. ..@ i : int [1:43367] 13 60 69 78 14 29 98 24 15 29 ...
      .. .. ..@ p : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
      .. .. ..@ Dim : int [1:2] 169 9835
      .. .. ..@ Dimnames:List of 2
      .. .. .. ..$ : NULL
      .. .. .. ..$ : NULL
      .. .. ..@ factors : list()
      ..@ itemInfo :'data.frame': 169 obs. of 3 variables:
      .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf"
        "ham" ...
      .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44
      44 44 44 44
      44 42 42 41 ...
      .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6
      6 6 6 6
      ...
      ..@ itemsetInfo:'data.frame': 0 obs. of 0 variables
```

10

这个数据集被结构化为一个稀疏矩阵对象，称为事务类。

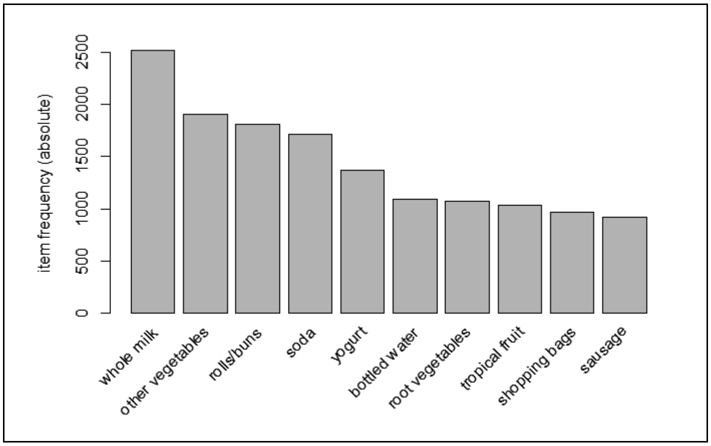
如果数据集结构是事务类型，那么标准的查看数据方法将会失效，但是arules包为我们提供了查看数据的其他方法。顺便说一下，如果你有一个数据框或矩阵想转换成事务类型，可以使用`as()`函数轻松完成。

> 下面的代码仅仅为了演示，不要实际运行：
>
> ```
> > # transaction.class.name <-
> as(current.data.frame,"transactions").
> ```

查看数据的最好方式是，使用arules包中的`itemFrequencyPlot()`函数生成项目频率图。你需要指定事务数据集、图中要显示的具有最高频率的项目数，以及要使用项目绝对频率还是相对频率。先使用绝对频率，看看频率最高的10个项目：

```
> itemFrequencyPlot(Groceries, topN = 10, type = "absolute")
```

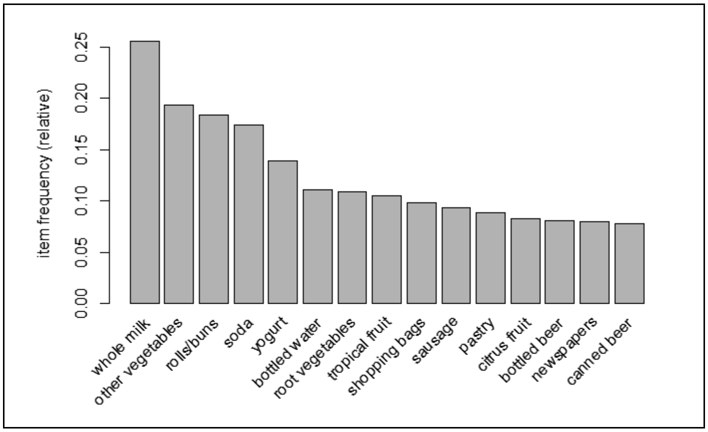
上述命令输出如下。



人们购买最多的项目是“全脂牛奶”，在9836条事务记录中占了大约2500条。如果想看看相对频率最高的前15个项目，可以使用下面的代码：

```
> itemFrequencyPlot(Groceries, topN = 15)
```

上述代码输出如下页图。



我们看到，啤酒在杂货店“最常被购买的商品”中只排在第13位（瓶装）和第15位（罐装）。只有不到10%的购买记录中包括瓶装啤酒和罐装啤酒。

仅为练习的话，知道这些就足够了，下面开始建模和评价。

10.4　模型构建与模型评价

我们从对全体关联规则的数据挖掘开始，然后再转到关于啤酒的特定规则。整个建模过程都使用Apriori算法，它使用arules包中的`apriori()`函数。使用这个函数时，需要确定的两个重要部分是数据集和参数组。对于参数组，你需要自己做出决定，确定最小支持度、置信度以及项集中的最大项数和最小项数。使用项目频率图和反复试错法，我们设定最小支持度为1/1000，最小置信度为90%。另外，设置项集中的最大项数为4。下面的代码可以生成一个对象，将其命名为`rules`：

```
> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf =
  0.9, maxlen=4))
```

调用这个对象，可以看到算法产生多少条关联规则：

```
    > rules
set of 67 rules
```

有多种方法可以查看规则。我极力推荐通过R基础包中的`options()`函数将数字的小数位设置为2，然后按照提升度由高到低排序，查看最前面的5条关联规则：

```
    > options(digits = 2)

> rules <- sort(rules, by = "lift", decreasing = TRUE)

> inspect(rules[1:5])
```

```
  lhs                 rhs                   support confidence lift
1 {liquor, red/blush wine}     => {bottled beer}      0.0019
      0.90 11.2
2 {root vegetables, butter, cream cheese }      => {yogurt}
      0.0010       0.91  6.5
3 {citrus fruit, root vegetables, soft cheese}=> {other vegetables}
      0.0010       1.00  5.2
4 {pip fruit, whipped/sour cream, brown bread}=> {other vegetables}
      0.0011       1.00  5.2
5 {butter,whipped/sour cream, soda} => {other vegetables}
      0.0013       0.93  4.8
```

请看，具有最高提升度的关联规则是，购买了白酒和红葡萄酒的顾客也很可能购买瓶装啤酒。我必须说这纯属巧合，绝对不是故意的。就像我经常说的，干得好不如干得巧。这条规则的支持度只有1.9/1000，说明这种购买行为并不常见。

也可以按照支持度和置信度排序。下面按照`by=confidence`降序排列，前5条关联规则如下所示：

```
    > rules <- sort(rules, by = "confidence", decreasing = TRUE)

> inspect(rules[1:5])
  lhs              rhs                 support confidence lift
1 {citrus fruit, root vegetables, soft cheese}=> {other vegetables}
     0.0010          1 5.2
2 {pip fruit, whipped/sour cream, brown bread}=> {other vegetables}
     0.0011          1 5.2
3 {rice, sugar} => {whole milk}          0.0012            1 3.9
4 {canned fish, hygiene articles} => {whole milk} 0.0011 1 3.9
5 {root vegetables, butter, rice} => {whole milk} 0.0010 1 3.9
```

由上可知，这些交易记录的置信度为100%。下面专门研究一下与啤酒有关的交易。可以用arules包中的`crossTable()`函数建立一个交叉表，然后按照需求进行研究。第一步就是根据数据集建立交叉表：

```
> tab <- crossTable(Groceries)
```

表格建立之后，检查商品之间的共同购买关系。先看看表格的前3行和前3列：

```
    > tab[1:3, 1:3]
            frankfurter sausage liver loaf
frankfurter         580      99          7
sausage              99     924         10
liver loaf            7      10         50
```

可以看到，在9835笔交易记录中，顾客们只购买了50次肝肠。此外，顾客购买了924次香肠时，有10次同时购买了肝肠。（我对肝肠的销量太失望了，只能这么计算才能使销量好看一点！）如果想专门看看某种商品，可以指定行号和列号，也可以简单地使用商品名：

```
    > table["bottled beer","bottled beer"]
[1] 792
```

有792条关于瓶装啤酒的交易记录。看看人们购买瓶装啤酒的同时，购买了多少次罐装啤酒：

```
    > table["bottled beer","canned beer"]
[1] 26
```

我早就预料到这个数字会很小，因为人们一般都是或者喝瓶装的啤酒，或者喝罐装的啤酒，不会同时喝两种啤酒。我就特别喜欢瓶装啤酒。如果遇到像占领华尔街那样的狂暴的抗议者，啤酒瓶还是保护自己的一件称手武器呢。

现在可以生成关于瓶装啤酒的关联规则了。依然使用`apriori()`函数，但这一次要加上参数appearance的设定。设定这个参数的意义是，我们需要关联规则的左侧项是能够提高瓶装啤酒购买率的那些项集，右侧项就是瓶装啤酒。在下面的代码中，请注意我调整了支持度和置信度的数值，你也可以根据需要随时调整：

```
    > beer.rules <- apriori(data = Groceries, parameter = list(support
      = 0.0015, confidence = 0.3), appearance = list(default = "lhs",
        rhs = "bottled beer"))
> beer.rules
set of 4 rules
```

只找出4条关联规则。我们已经看过其中一条，下面看看其他3条规则，按照提升度降序排列：

```
    > beer.rules <- sort(beer.rules, decreasing = TRUE, by = "lift")

> inspect(beer.rules)
  lhs                      rhs            support confidence lift
1 {liquor, red/blush wine} => {bottled beer} 0.0019 0.90 11.2
2 {liquor}                 => {bottled beer}   0.0047 0.42   5.2
3 {soda, red/blush wine} => {bottled beer}   0.0016 0.36   4.4
4 {other vegetables, red/blush wine} => {bottled beer}0.0015 0.31
      3.8
```

在所有规则中，对瓶装啤酒的购买都与酒类有关，不是白酒就是红葡萄酒，这也没什么可惊讶的。有趣的是，白葡萄酒不在其中。下面更深入地看看瓶装啤酒和不同种类的葡萄酒之间的共同购买关系：

```
    > tab["bottled beer", "red/blush wine"]
[1] 48

> tab["red/blush wine", "red/blush wine"]
[1] 189

> 48/189
[1] 0.25

> tab["white wine", "white wine"]
[1] 187

> tab["bottled beer", "white wine"]
[1] 22
```

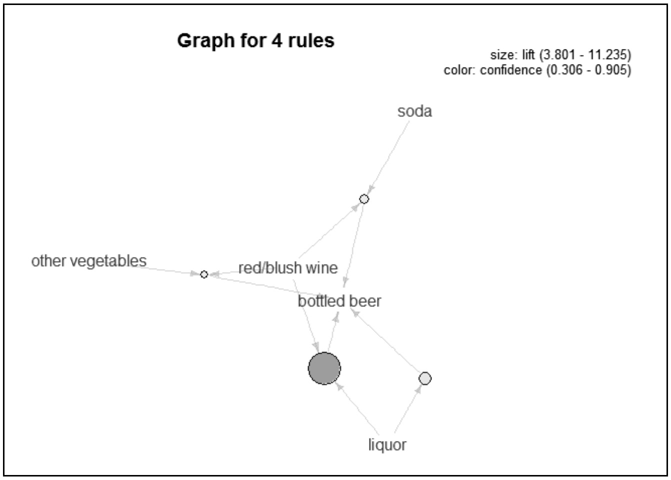
```
> 22/187
[1] 0.12
```

这是个很有趣的事实，购买红葡萄酒时，有25%的顾客会同时购买瓶装啤酒；但购买白葡萄酒时，只有12%的顾客会同时购买瓶装啤酒。当然，我们不知道为什么会这样，但这可以帮助我们确定如何在杂货店中摆放产品。进行下一部分内容之前，看看关联规则的统计图，这是使用arulesViz包中的`plot()`函数实现的。

图形有很多选项可以设置，在本例中，我们设定需要点线图，显示关联规则提升度，并且用颜色深浅表示置信度。如下所示：

```
> plot(beer.rules, method = "graph", measure = "lift", shading =
"confidence")
```

上述命令输出如下。



由上图可知，“白酒/红葡萄酒”提供了最高提升度和置信度，分别由圆点的大小和颜色深浅表示。

在这个简单的案例分析中，我们展示了使用R进行购物篮分析的便捷性。购物篮分析可以和很多其他统计分析技术一起使用，比如客户细分、纵向购买历史数据分析等，它们可以用于广告展示、协同促销等商业领域。下面，我们进一步讨论顾客对商品进行评价的情况，学习如何构建和测试推荐引擎。

10.5　推荐引擎简介

我们将集中讨论用户对以前浏览过或购买过的商品进行排名或评价的情况。推荐系统的设计

有两种主要方式：**协同过滤**和**基于内容的推荐**（Ansari、Essegaier与Kohli，2000）。我们主要讨论前一种方式，因为这就是将要使用的R包recommenderlab的主要功能。

至于基于内容的推荐方法，它的理念是将用户偏好与商品属性联系在一起。对于电影和电视剧推荐来说，商品属性就是体裁、演员表和故事情节等。使用这种方法的话，推荐完全是基于用户的评价给出的，与其他任何人的评价都没有关系。基于内容的推荐方法的优点是：加入一个新项目时，它如果与某个用户档案中的兴趣偏好相符，就会被推荐给这个用户，而不必等到其他用户对它进行评价之后（这就是所谓的“第一评价”问题）。但是，基于内容的推荐方法在可用内容比较少时会遇到问题，有时是领域问题，有时发生在新用户进入系统的时候。这样就会导致不唯一的推荐，也就是糟糕的推荐。（Lops、Gemmis与Semeraro，2011）

在协同过滤推荐方法中，推荐是基于数据库中一些或全部人员提供的多条评价进行的。从本质上来说，这种方法利用的是群体的智慧。

对于协同过滤，我们重点讨论以下4种方法：

- **基于用户的协同过滤**
- **基于项目的协同过滤**
- **奇异值分解**
- **主成分分析**

讨论实际的商业案例之前，先简单介绍这几种方法。还要注意，recommenderlab包不是作为一种实际的实现工具来设计开发的，它是一种实验工具，你既可以使用它研究recommenderlab包提供的算法，也可以实验你自己想要研究的其他算法。

10.5.1 基于用户的协同过滤

在基于用户的协同过滤算法中，**先找到与用户相似的近邻，然后将这些近邻的评价综合起来产生一个推荐，将其中未被用户发现的部分推荐给用户**（Hahsler，2011）。可以通过KNN找出与我们要推荐的用户最相似的近邻，也可以使用带有最小阈值的其他相似度测量方式。recommenderlab包提供了两种相似度测量方式：**皮尔逊相关系数**和**余弦相似度**。我不介绍这些测量方式的公式，因为recommenderlab包的说明文档中已经讲得很清楚了。

确定了寻找近邻的方式之后，算法通过计算相似度确定近邻。计算相似度时，只使用该用户与近邻共同评价了的项目。然后通过某种评分方式（比如简单地求平均数），将近邻的评价综合起来，为用户预测某个项目上的评分。

来看一个简单的例子。下表共有6个独立用户对4部电影进行了评价，只有我没有评价《疯狂的麦克斯》。设$k=1$，那么我的最近邻就是Homer，Bart次之；尽管在不喜欢《复仇者联盟》这一点上，Flanders和我一模一样。所以，使用Homer对《疯狂的麦克斯》的评价（4分）来预测我对

这部电影的评分也是4分。

	《复仇者联盟》	《美国狙击手》	《悲惨世界》	《疯狂的麦克斯》
Homer	3	5	3	4
Marge	5	2	5	3
Bart	5	5	1	4
Lisa	5	1	5	2
Flanders	1	1	4	1
我	1	5	2	?

有很多方式可以对数据进行加权和控制偏差。举例来说，Flanders和其他用户相比特别喜欢打低分，所以要对他的打分进行规范化，他在某个项目上的新评分等于原评分减去他在所有项目上的评分的均值。

基于用户的协同过滤算法的缺点是，为了计算所有用户的相似度，整个数据库必须驻留在内存中，这在计算能力和时间上都是一笔很大的开销。

10.5.2　基于项目的协同过滤

你可能已经猜到了，基于项目的协同过滤算法在进行推荐时使用的是项目之间的相似度，而不是用户之间的相似度。**这种方法背后的假设是用户更倾向于那些和他们喜欢的其他项目类似的项目**（Hahsler，2011）。模型是通过计算所有项目之间的两两相似度建立起来的。常用的相似度测量方式是皮尔逊相关系数和余弦相似度。为了减小相似度矩阵的规模，可以设定只使用k个最相似的项目。但是，对近邻数目的限制会明显降低正确率，导致基于项目的协同过滤算法的性能劣于基于用户的协同过滤算法。

还是看一下前面那个简单的例子，从下表可知，如果$k = 1$，那么与《疯狂的麦克斯》最相似的项目是《美国狙击手》。于是，可以使用我对《美国狙击手》的评价来预测我对《疯狂的麦克斯》的评价，如下所示。

	《复仇者联盟》	《美国狙击手》	《悲惨世界》	《疯狂的麦克斯》
Homer	3	5	3	4
Marge	5	2	5	3
Bart	5	5	1	4
Lisa	5	1	5	2
Flanders	1	1	4	1
我	1	5	2	?

10.5.3　奇异值分解和主成分分析

现在，包含几百万用户和项目的数据集已经很常见了。尽管评价矩阵没有那么大，但降低维

度还是有好处的。降维的方法是，建立一个更小的，但能反映高维矩阵中大部分信息的低维矩阵。这样可能会使你发现数据中重要的潜在因子和相应的权重，这些因子也许会揭示一些评价矩阵中的重要信息，比如电影体裁或书籍主题。尽管你可能无法辨别有意义的因子，但降维技术也可以过滤数据中的噪声。

大数据集的一个问题是，你很可能得到一个稀疏矩阵，其中很多评价是空白的。降维技术的缺点是不能支持带有缺失值的矩阵，必须进行数据填补。有很多种技术可以用来尝试完成数据填补任务，比如使用均值、中位数或0来代替缺失值。recommenderlab包中的默认方式是使用中位数。

什么是奇异值分解？简单地说，它就是一种矩阵分解方法，有助于将一组关联特征转换为不关联的特征。假设有矩阵A，这个矩阵可以分解为3个矩阵：U、D和V^T。U是一个正交矩阵，D是一个半正定对角矩阵，V^T是一个正交矩阵的转置。现在看一下评价矩阵，并使用R演示一个例子。

首先，重新建立评价矩阵（将它看作矩阵A，如下代码所示）：

```
    > ratings <- c(3, 5, 5, 5, 1, 1, 5, 2, 5, 1, 1, 5, 3, 5, 1, 5, 4
      ,2, 4, 3, 4, 2, 1, 4)

> ratingMat <- matrix(ratings, nrow = 6)

> rownames(ratingMat) <- c("Homer", "Marge", "Bart", "Lisa",
      "Flanders", "Me")

> colnames(ratingMat) <- c("Avengers", "American Sniper", "Les
      Miserable", "Mad Max")

> ratingMat
Avengers  American Sniper  Les Miserable  Mad  Max
Homer        3               5             3     4
Marge        5               2             5     3
Bart         5               5             1     4
Lisa         5               1             5     2
Flanders     1               1             4     1
Me           1               5             2     4
```

然后，使用R基础包中的`svd()`函数，将评价矩阵分解为上面所说的3个矩阵，R将其分别命名为$d、$u和$v。可以认为$u中的值就是某个用户在相应因子上的载荷，$v中的值是某个电影在相应维度上的载荷，例如，《疯狂的麦克斯》在维度1上的载荷就是−0.116：

```
    > svd <- svd(ratingMat)

> svd
$d
[1] 16.1204848   6.1300650   3.3664409   0.4683445
```

```
$u
           [,1]       [,2]        [,3]       [   ,4]
[1,] -0.4630576  0.2731330   0.2010738 -0.27437700
[2,] -0.4678975 -0.3986762  -0.0789907  0.53908884
[3,] -0.4697552  0.3760415  -0.6172940 -0.31895450
[4,] -0.4075589 -0.5547074  -0.1547602 -0.04159102
[5,] -0.2142482 -0.3017006   0.5619506 -0.57340176
[6,] -0.3660235  0.4757362   0.4822227  0.44927622

$v
           [,1]       [,2]        [,3]       [,4]
[1,] -0.5394070 -0.3088509  -0.77465479 -0.1164526
[2,] -0.4994752  0.6477571   0.17205756 -0.5489367
[3,] -0.4854227 -0.6242687   0.60283871 -0.1060138
[4,] -0.4732118  0.3087241   0.08301592  0.8208949
```

可以很容易地算出通过降维能够解释多少方差。先将矩阵$d的对角线上的数字相加，然后看看只用两个因子时能够解释多少方差。如下所示：

```
    > sum(svd$d)
[1] 26.08534

> var <- sum(svd$d[1:2])

> var
[1] 22.25055

> var/sum(svd$d)
[1] 0.8529908
```

使用4个因子中的两个，能够解释整个矩阵85%的全部方差。可以算出降维后产生的评分，只需编写一个函数。（特别感谢www.stackoverflow.com上的回复者，在他们的帮助下我才能实现这个函数。）这个函数允许我们指定预测中要使用的因子数量，它将$u、$v和$d相乘来算出评分：

```
    > f1 <- function(x) {
score = 0
for(i in 1:n )
score <- score + svd$u[,i] %*% t(svd$v[,i]) * svd$d[i]
return(score)}
```

设$n = 4$，调用这个函数，重建初始的评价矩阵：

```
    > n = 4

> f1(svd)
     [,1] [,2] [,3] [,4]
[1,]    3    5    3    4
[2,]    5    2    5    3
[3,]    5    5    1    4
[4,]    5    1    5    2
[5,]    1    1    4    1
[6,]    1    5    2    4
```

同样，设$n = 2$，看看输出的结果：

```
> n = 2

> f1(svd)
            [,1]      [,2]     [,3]     [,4]
[1,] 3.509402 4.8129937 2.578313 4.049294
[2,] 4.823408 2.1843483 5.187072 2.814816
[3,] 3.372807 5.2755495 2.236913 4.295140
[4,] 4.594143 1.0789477 5.312009 2.059241
[5,] 2.434198 0.5270894 2.831096 1.063404
[6,] 2.282058 4.8361913 1.043674 3.692505
```

通过奇异值分解可以降低数据维度，并且可能发现有意义的潜在因子。

你完成了前一章就能知道，主成分分析和奇异值分解具有相似的作用。实际上，这两种技术连接非常紧密，经常交互使用，因为它们都用到了矩阵分解技术。那么，它们之间的区别是什么？简言之，主成分分析是基于协方差矩阵的，这个矩阵是对称的。要进行主成分分析，需要先从数据开始，对数据进行中心化，然后计算协方差矩阵并进行对角化，最后生成主成分。

对本例中的数据应用第9章中的部分主成分分析代码，看看有什么不同：

```
> library(psych)

> pca <- principal(ratingMat, nfactors = 2, rotate = "none")

> pca
Principal Components Analysis
Call: principal(r = ratingMat, nfactors = 2, rotate =
"none")
Standardized loadings (pattern matrix) based upon correlation
    matrix
                  PC1   PC2   h2    u2
Avengers        -0.09  0.98 0.98 0.022
American Sniper  0.99 -0.01 0.99 0.015
Les Miserable   -0.90  0.18 0.85 0.150
Mad Max          0.92  0.29 0.93 0.071

                        PC1  PC2
SS loadings           2.65 1.09
Proportion Var        0.66 0.27
Cumulative Var        0.66 0.94
Proportion Explained  0.71 0.29
Cumulative Proportion 0.71 1.00
```

可以看出，主成分分析更容易解释。《美国狙击手》和《疯狂的麦克斯》在第一个主成分上具有高载荷，在第二个主成分上，只有《复仇者联盟》具有高载荷。此外，这两个主成分能够解释94%的数据总体方差。

使用简单的评价矩阵介绍协同过滤技术之后，下面看看更复杂的使用真实数据的案例。

10.6　推荐系统的业务理解

从字面上看，这个商业案例是个笑话。更确切地说，是一大波笑话，因为我们要使用recommenderlab包中的Jester5k数据集。这个数据集包含了5000个用户对100个笑话的评价，这些数据来自"小丑在线笑话推荐系统"（Jester Online Joke Recommender System）。数据收集时间是1999年4月~2003年5月，所有用户都至少评价了36个笑话（Goldberg、Roeder、Gupta与Perkins，2001）。我们的目标是比较各种推荐算法，然后找出最好的那种。

既然如此，我认为非常有必要从一个关于统计学家的笑话开始，好让我们进入状态。我不确定如何对这个笑话进行恰当的评价，但它的确风靡了整个互联网。

一位统计学家的太太生了一对双胞胎，丈夫乐坏了。他打电话给牧师，牧师也很高兴。"周六把他们带到教堂来，我要给他们举行洗礼"，牧师说。"不行，"统计学家答道，"你只能洗一个，另一个我要用作对照组。"

10.7　推荐系统的数据理解与数据准备

对于这个练习，使用recommenderlab即可。这个程序包是由南卫理工会大学莱尔工程实验室开发的，他们还有一个很棒的站点，上面有这个程序包的支持文档（参见https://lyle.smu.edu/IDA/recommenderlab/）：

```
    > library(recommenderlab)

> data(Jester5k)

> Jester5k
5000 x 100 rating matrix of class 'realRatingMatrix' with
362106 ratings.
```

评价矩阵包含362 106个评价。想看一个用户的评价列表非常容易，我们看看第10个用户的评价。下面的输出只列出对前5个笑话的评价，省略了其他：

```
    > as(Jester5k[10,], "list")
$u12843
   j1    j2    j3    j4    j5 ...
-1.99 -6.89  2.09 -4.42 -4.90 ...
```

还可以看看某个用户（用户10）的平均评价和某个笑话（笑话1）的平均评价，如下所示：

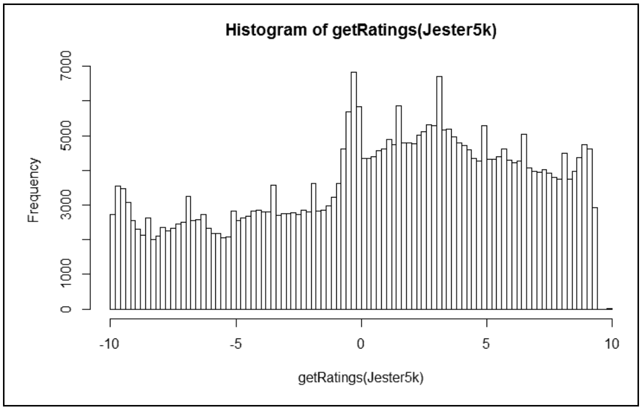
```
    > rowMeans(Jester5k[10,])
u12843
  -1.6

> colMeans(Jester5k[,1])
  j1
0.92
```

数据理解的更好方式是画出这些评价的直方图，应该查看原始数据和规范化后数据的图。使用recommenderlab包中的`getRating()`函数生成统计图：

```
> hist(getRatings(Jester5k), breaks=100)
```

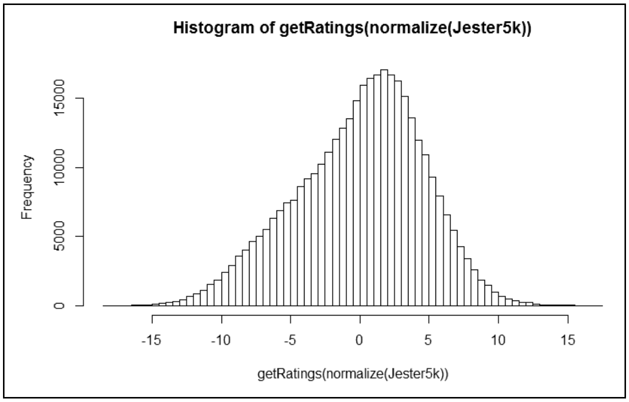
上述命令输出如下。



程序包中的`normalize()`函数可以对数据进行中心化，即用笑话的评价减去该笑话所有评价的均值。从上图可以看到，评价的分布稍稍偏向正的一侧，规范化后的数据也反映出这个趋势。可以看到结果非常像正态分布，但依然稍稍偏向正的一侧，如下所示：

```
> hist(getRatings(normalize(Jester5k)), breaks = 100)
```

上述命令输出如下。



进行模型构建和模型评价之前，可以先使用recommenderlab包中的`evaluationScheme()`函数轻松建立训练集和测试集。我们按照80/20的比例划分训练集数据和测试集数据。如果你愿意，也可以选择*k*折交叉验证和自助抽样法。此外还要设定，对于测试集数据，算法使用15个评

价进行预测，其余的评价项目用于计算误差。另外，还要设定“好评价”的阈值，本例设为大于等于5：

```
    > set.seed(123)

> e <- evaluationScheme(Jester5k, method="split",
train=0.8, given=15, goodRating=5)

>  e
Evaluation scheme with 15 items given
Method: 'split' with 1 run(s).
Training set proportion: 0.800
Good ratings: >=5.000000
Data set: 5000 x 100 rating matrix of class
'realRatingMatrix' with 362106
 ratings.
```

建立训练集数据和测试集数据之后，开始构建模型，并对不同的推荐技术进行评价：基于用户的推荐、基于项目的推荐、基于流行度的推荐、奇异值分解、主成分分析以及随机推荐。

10.8　推荐系统的建模与评价

如果要建立与测试推荐引擎，可以使用同一个函数Recommender()，只须为每种推荐技术改变设置即可。如果想看看这个函数的具体功能和对应于6种推荐技术的全部参数设置，可以查看其注册信息。请看下面注册信息中的基于物品的协同过滤算法，可以知道默认设置是使用余弦相似度寻找30个近邻，并且对数据进行中心化，缺失数据不用0进行填补：

```
    > recommenderRegistry$get_entries(dataType =
"realRatingMatrix")

$ALS_realRatingMatrix
    Recommender method: ALS for realRatingMatrix
    Description: Recommender for explicit ratings based on latent
      factors, calculated by alternating least squares algorithm.
    Reference: Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, Rong
      Pan (2008).
    Large-Scale Parallel Collaborative Filtering for the Netflix Prize,
      4th Int'l
    Conf. Algorithmic Aspects in Information and Management, LNCS 5034.
    Parameters:
    normalize lambda n_factors n_iterations min_item_nr seed
    1 NULL 0.1 10 10 1 NULL

    $ALS_implicit_realRatingMatrix
    Recommender method: ALS_implicit for realRatingMatrix
    Description: Recommender for implicit data based on latent factors,
    calculated by alternating least squares algorithm.
    Reference: Yifan Hu, Yehuda Koren, Chris Volinsky (2008).
      Collaborative
```

```
Filtering for Implicit Feedback Datasets, ICDM '08 Proceedings of
  the 2008
Eighth IEEE International Conference on Data Mining, pages 263-272.
Parameters:
lambda alpha n_factors n_iterations min_item_nr seed
1 0.1 10 10 10 1 NULL

$IBCF_realRatingMatrix
Recommender method: IBCF for realRatingMatrix
Description: Recommender based on item-based collaborative
  filtering.
Reference: NA
Parameters:
k method normalize normalize_sim_matrix alpha na_as_zero
1 30 "Cosine" "center" FALSE 0.5 FALSE

$POPULAR_realRatingMatrix
Recommender method: POPULAR for realRatingMatrix
Description: Recommender based on item popularity.
Reference: NA
Parameters:
  normalize aggregationRatings aggregationPopularity
  1 "center" new("standardGeneric" new("standardGeneric"

$RANDOM_realRatingMatrix
Recommender method: RANDOM for realRatingMatrix
Description: Produce random recommendations (real ratings).
Reference: NA
Parameters: None

$RERECOMMEND_realRatingMatrix
Recommender method: RERECOMMEND for realRatingMatrix
Description: Re-recommends highly rated items (real ratings).
Reference: NA
Parameters:
  randomize minRating
1 1 NA

$SVD_realRatingMatrix
Recommender method: SVD for realRatingMatrix
Description: Recommender based on SVD approximation with column-mean
imputation.
Reference: NA
Parameters:
 k maxiter normalize
 1 10 100 "center"

$SVDF_realRatingMatrix
Recommender method: SVDF for realRatingMatrix
Description: Recommender based on Funk SVD with gradient descend.
Reference: NA
Parameters:
 k gamma lambda min_epochs max_epochs min_improvement normalize
 1 10 0.015 0.001 50 200 1e-06 "center"
```

```
verbose
1 FALSE

$UBCF_realRatingMatrix
Recommender method: UBCF for realRatingMatrix
Description: Recommender based on user-based collaborative
  filtering.
Reference: NA
Parameters:
 method nn sample normalize
 1 "cosine" 25 FALSE "center"
```

下面是在训练数据上使用这6种推荐技术的方法。为简单起见，全部使用算法的默认设置。你也可以调整参数设置，方法很简单，将要改变的参数作为一个列表放在函数中即可：

```
    > ubcf <- Recommender(getData(e,"train"), "UBCF")

> ibcf <- Recommender(getData(e,"train"), "IBCF")

> svd <- Recommender(getData(e, "train"), "SVD")

> popular <- Recommender(getData(e, "train"), "POPULAR")

> pca <- Recommender(getData(e, "train"), "PCA")

> random <- Recommender(getData(e, "train"), "RANDOM")
```

现在，通过`predict()`函数和`getData()`函数，分别使用6种技术在测试集上使用15个项目进行预测，如下所示：

```
    > user_pred <- predict(ubcf, getData(e, "known"), type = "ratings")

> item_pred <- predict(ibcf, getData(e, "known"), type = "ratings")

> svd_pred <- predict(svd, getData(e, "known"), type = "ratings")

> pop_pred <- predict(popular, getData(e, "known"), type =
     "ratings")

> rand_pred <- predict(random, getData(e, "known"), type =
     "ratings")
```

使用`calcPredictionAccuracy()`函数计算预测值和测试集未知部分的误差。计算结果包括所有方法的均方根误差（RMSE）、均方误差（MSE）和平均绝对误差（MAE）。先单独看一下基于用户的协同过滤的误差。建立所有6种方法的对象之后，使用`rbind()`函数建立一个表格，并使用`rowname()`函数为表格的每一行命名：

```
    > P1 <- calcPredictionAccuracy(user_pred, getData(e,
"unknown"))

> P1
```

```
RMSE  MSE  MAE
4.5 19.9  3.5

> P2 <- calcPredictionAccuracy(item_pred, getData(e, "unknown"))

> P3 <- calcPredictionAccuracy(svd_pred, getData(e, "unknown"))

> P4 <- calcPredictionAccuracy(pop_pred, getData(e, "unknown"))

> P5 <- calcPredictionAccuracy(rand_pred, getData(e, "unknown"))

> error <- rbind(P1, P2, P3, P4, P5)

> rownames(error) <- c("UBCF", "IBCF", "SVD", "Popular", "Random")

> error
         RMSE MSE MAE
    UBCF      4.5 20  3.5
    IBCF      4.6 22  3.5
    SVD       4.6 21  3.7
    Popular   4.5 20  3.5
    Random    6.3 40  4.9
```

在输出结果中可以看到，基于用户的推荐和基于流行度的推荐技术表现得比基于物品的协同过滤和奇异值分解稍好一点，但它们都优于随机推荐。

还有一种比较方式是使用`evaluate()`函数。使用`evaluate()`进行比较时，可以使用其他方式查看各种技术的表现，比如使用图形。因为基于用户的协同过滤和基于流行度的推荐是表现最佳的两种算法，我们就看看这两种算法与基于物品的协同过滤算法三者之间的比较。

首先建立一个要比较的算法的列表，如下所示：

```
    > algorithms <- list(POPULAR = list(name = "POPULAR"),
    UBCF =list(name = "UBCF"), IBCF = list(name = "IBCF"))

> algorithms
$POPULAR
$POPULAR$name
[1] "POPULAR"

$UBCF
$UBCF$name
[1] "UBCF"

$IBCF
$IBCF$name
[1] "IBCF"
```

在本例中，我们比较前5、10、15个笑话推荐：

```
    > evlist <- evaluate(e, algorithms, n = c(5, 10, 15))
POPULAR run
```

```
 1  [0.07sec/4.7sec]
UBCF run
 1  [0.04sec/8.9sec]
IBCF run
 1  [0.45sec/0.32sec]3
```

请注意，执行上面的命令可以知道每种算法的运行时间。现在通过avg()函数检查各种技术的表现：

```
    > set.seed(1)

    > avg(evlist)
$POPULAR
         TP     FP     FN     TN    precision    recall   TPR    FPR
  5    2.07   2.93   12.9   67.1        0.414     0.182 0.182 0.0398
  10   3.92   6.08   11.1   63.9        0.393     0.331 0.331 0.0828
  15   5.40   9.60    9.6   60.4        0.360     0.433 0.433 0.1314

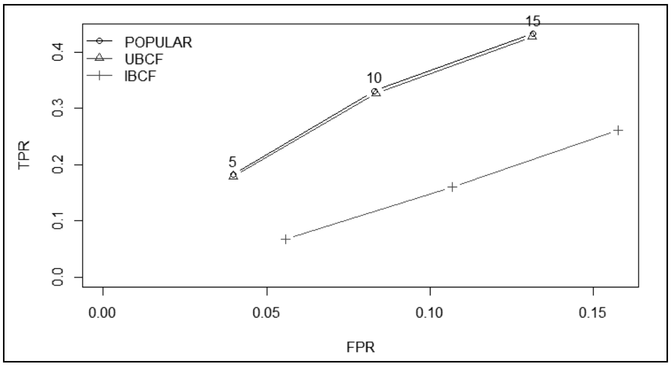
$UBCF
         TP     FP     FN     TN    precision    recall   TPR    FPR
  5    2.07   2.93  12.93   67.1        0.414     0.179 0.179 0.0398
  10   3.88   6.12  11.11   63.9        0.389     0.326 0.326 0.0835
  15   5.41   9.59   9.59   60.4        0.360     0.427 0.427 0.1312

$IBCF
         TP     FP     FN     TN    precision    recall     TPR    FPR
  5    1.02   3.98   14.0   66.0        0.205    0.0674 0.0674 0.0558
  10   2.35   7.65   12.6   62.4        0.235    0.1606 0.1606 0.1069
  15   3.72  11.28   11.3   58.7        0.248    0.2617 0.2617 0.1575
```

请注意，基于流行度的方法和基于用户的协同过滤算法的表现几乎没有什么差别。可能有人会说，如果进行模型选择，那么简单易行的基于流行度的算法可能更好。我们可以绘制**受试者工作特征曲线**比较结果，在图中你可以比较TPR和FPR，或者精确度/召回度。如下所示：

```
> plot(evlist, legend = "topleft", annotate = TRUE)
```

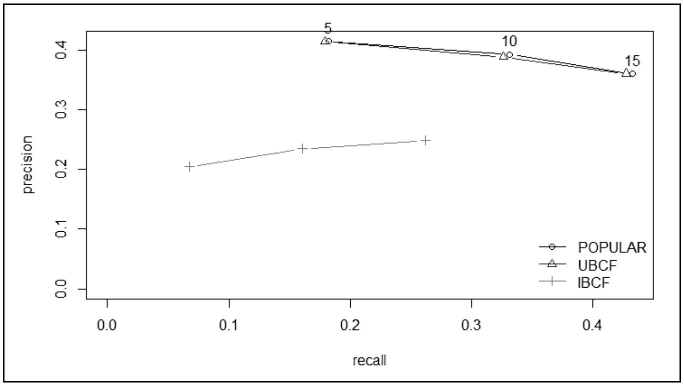
上述命令输出如下。



要得到精确度/召回度曲线图，只需在`plot()`函数中指定`"prec"`即可：

```
> plot(evlist, "prec", legend = "bottomright", annotate = TRUE)
```

上述命令输出如下。



在图中可以清楚地看到，基于流行度和基于用户的推荐算法几乎是等同的，并且都优于基于项目的算法。参数`annotate = TRUE`的作用是在点的旁边显示数字，图中每条线上的3个点对应模型评价中使用的3种推荐数量。

用统计图做模型评价非常简单，但是对于一个具体用户来说，模型会给出什么样的推荐呢？通过一些简单的编码即可知道答案。首先，在整个数据集上建立基于流行度的推荐引擎，然后找出为前2个用户做出的前5个推荐。在整个数据集上使用`recommend()`函数，如下所示：

```
    > R1 <- Recommender(Jester5k, method = "POPULAR")

> R1
Recommender of type 'POPULAR' for 'realRatingMatrix'
learned using 5000 users.
```


现在，只需得到为前2个用户做出的前5个推荐，并转换为一个列表：

```
    > recommend <- predict(R1, Jester5k[1:2], n = 5)

> as(recommend, "list")
$u2841
    [1] "j89" "j72" "j76" "j88" "j83"

    $u15547
[1] "j89" "j93" "j76" "j88" "j91"
```

还可以看到一个用户对每个笑话的评价分数，在`predict()`函数中进行相应设置，然后将结果放入一个矩阵以供查看即可。查看10位用户（用户300~用户309）对3个笑话（笑话71~笑话73）的评价：

```
    > rating <- predict(R1, Jester5k[300:309], type = "ratings")

> rating
10 x 100 rating matrix of class 'realRatingMatrix' with 322
ratings.

> as(rating, "matrix")[, 71:73]
           j71  j72     j73
    u7628  -2.042 1.50 -0.2911
    u8714      NA   NA      NA
    u24213 -2.935      NA -1.1837
    u13301  2.391    5.93  4.1419
    u10959     NA      NA      NA
    u23430 -0.432    3.11      NA
    u11167 -1.718    1.82  0.0333
    u4705  -1.199    2.34  0.5519
    u24469 -1.583    1.96  0.1686
    u13534 -1.545    2.00      NA
```

矩阵中的数字表示预测出的用户对笑话的评价分数，`NA`表示用户没有对该笑话进行评价。

最后介绍如何在二值评价（好/坏、1/0）的情况下建立推荐引擎。需要将评价分数转换成二值形式，大于等于5的评价记为1，小于5的评价记为0。使用Recommenderlab包中的`binarize()`函数，设定`minRating=5`：

```
> Jester.bin <- binarize(Jester5k, minRating = 5)
```

现在，为了满足算法训练的需要，我们要找出那些具有一定数量的评价为1的记录。为了方便说明，假设这个数量大于10。要建立满足需要的数据子集，可以使用下面的代码：

```
    > Jester.bin <- Jester.bin[rowCounts(Jester.bin) > 10]

> Jester.bin
3054 x 100 rating matrix of class 'binaryRatingMatrix' with 84722
      ratings.
```

还需要建立evaluationScheme。在本例中，使用参数`cross-validation`，表示使用交叉验证。默认设置是使用10折交叉验证，为了方便，我们设定$k = 5$，这样会减少运行时间：

```
    > set.seed(456)

> e.bin <- evaluationScheme(Jester.bin, method = "crossvalidation",
      k = 5, given = 10)
```

为了进行比较，算法列表包括随机推荐、基于流行度的推荐和基于用户的推荐：

```
> algorithms.bin <- list("random" = list(name = "RANDOM", param =
  NULL), "popular" = list(name = "POPULAR", param = NULL), "UBCF" =
  list(name = "UBCF"))
```

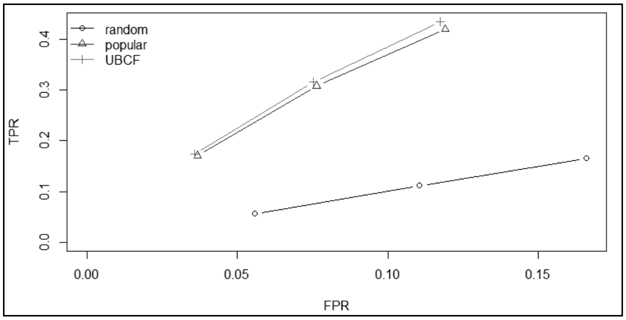
现在可以建立模型了，如下所示：

```
> results.bin <- evaluate(e.bin, algorithms.bin, n = c(5, 10, 15))
RANDOM run
1 [0sec/0.41sec]
2 [0.01sec/0.39sec]
3 [0sec/0.39sec]
4 [0sec/0.41sec]
5 [0sec/0.4sec]
POPULAR run
1 [0.01sec/3.79sec]
2 [0sec/3.81sec]
3 [0sec/3.82sec]
4 [0sec/3.92sec]
5 [0.02sec/3.78sec]
UBCF run
1 [0sec/5.94sec]
2 [0sec/5.92sec]
3 [0sec/6.05sec]
4 [0sec/5.86sec]
5 [0sec/6.09sec]
```

忽略性能比较表格，直接看统计图：

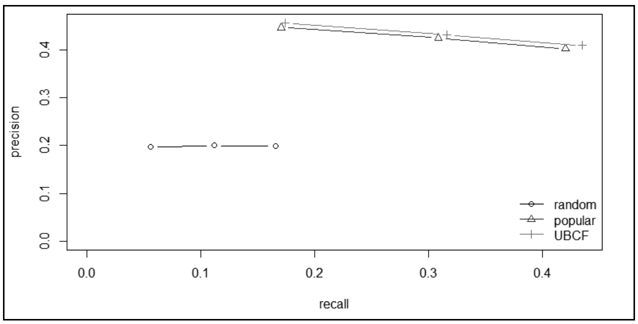
```
> plot(results.bin, legend = "topleft")
```

上述命令输出如下。




```
> plot(results.bin, "prec", legend = "bottomright")
```

上述命令输出如下。



基于用户的算法比基于流行度的算法稍稍好一点，但你可以看出，二者明显优于随机推荐。在这个案例中，应该由决策团队最后决定使用哪种算法。

10.9　序列数据分析

> "有些已知的已知，也就是我们知道我们知道。有些已知的未知，也就是我们知道有些事我们不知道。但是，也有一些未知的未知，就是那些我们不知道我们不知道的事。"
>
> ——唐纳德·拉姆斯菲尔德，美国前国防部长

本书出版前，我遇到的一个商业问题就是围绕产品序列分析进行的。我的团队使用复杂的Excel表格和透视表，以及一大堆SAS代码来从中分析有用的知识。遇到这个问题之后，我研究了如何用R解决问题。令人惊喜的是，我误打误撞地发现了TraMineR这个专门为解决这种问题而设计的软件包。我相信，使用R进行序列分析会大大简化分析过程。

这个软件包是为社会科学设计的，但适用于所有要挖掘和学习观测状态如何随着固定区间或事件（纵向数据）而变化的情况。前面提到的情况就是它的一个经典应用，在那个应用中，你想弄清楚顾客购买商品的顺序。这有利于实现一个推荐引擎，在这个引擎中，你可以生成下一次购买行为（也有人将其称为下一次合理的产品供应）的概率。另一个应用案例发生在卫生保健领域，用来研究患者接受治疗或药物的顺序，甚至还可以研究医生开处方的习惯。我从事过这方面的工作，当时创建了各种简单或复杂的马尔科夫链来建立模型并进行预测。实际上，TraMineR也可以创建马尔科夫链转移矩阵来支持这种模型。

我们要检查的代码会完成很多艰苦的工作，包括创建、计数以及绘制随时间变化的各种转换组合，还要包含协变量。这就是我们的重点，但请记住，也可以建立一个相异度矩阵来进行聚类分析。在这个实操训练中，核心特征由以下几个指标组成：

- 转换率
- 每种状态的持续时间
- 序列频率

让我们开始吧。

序列分析应用

为了进行这个练习，我自己创造了一个数据集，你可以从GitHub上下载：

https://github.com/datameister66/data/blob/master/sequential.csv

在TraMineR包中也可以找到数据集和教程，但我的意图是创造一些新的东西来反映我遇到

的情形。这个数据集几乎都是随机生成的（加入了一点人工干预），所以不要用它套用任何真实世界中的数据。该数据集包含5000条观测，每条观测都是某位顾客的购买历史记录，还包含9个变量。

- ❑ `Cust_segment`：一个因子变量，表示顾客分类（参见第8章）。
- ❑ 8个离散型购买事件，分别为`Purchase1~Purchase8`：请记住，它们不是基于时间序列的事件，也就是说，顾客可以同时购买所有8种商品，但要有一定的顺序。

每个购买变量中保存着购买商品的名称，共有7种商品，ProductA~ProductG。它们是些什么商品呢？这并不重要！你可以根据实际情况尽情发挥想象力。如果一个顾客只购买了一件商品，那么Purchase1就会包含这件商品的名称，其他变量都是NULL。

将这个文件加载到数据框中，为了使版面更加整洁清晰，我简化了结构函数的输出：

```
> df <- read.csv("sequential.csv")

> str(df)
'data.frame': 5000 obs. of 9 variables:
$ Cust_Segment: Factor w/ 4 levels "Segment1","Segment2",..: 1 1 1
1 1 1 1 1 1 1 ...
$ Purchase1 : Factor w/ 7 levels "Product_A","Product_B",..: 1 2 7
3 1 4 1 4 4 4 ...
```

下面进行一些数据探索，首先使用表格表示各个顾客分类中的顾客数量，以及第一次商品购买的分类数量：

```
> table(df$Cust_Segment)

Segment1 Segment2 Segment3 Segment4
    2900      572      554      974

> table(df$Purchase1)

Product_A Product_B Product_C Product_D Product_E Product_F
Product_G
     1451       765       659      1060       364       372
329
```

Segment1是最大的顾客分类，首件商品中，购买最多的是ProductA。但在所有商品购买中，它是购买次数最多的吗？以下代码可以告诉我们答案：

```
> table(unlist(df[, -1]))

Product_A Product_B Product_C Product_D Product_E Product_F
Product_G
3855      3193      3564      3122      1688      1273  915
22390
```

没错，ProductA就是购买次数最多的商品。NULL值的数量是22 390。

你可能想知道，我们能否不费力气就完成一些摘要统计。确实可以。下面对dplyr包中的count()函数和arrange()函数进行了充分利用，检查第一次购买行为和第二次购买行为之间的序列频率：

```
> dfCount <- count(df, Purchase1, Purchase2)

> dfCount <- arrange(dfCount, desc(n))

> dim(dfCount)
[1] 56 3

> head(dfCount)
Source: local data frame  [6 x 3]
Groups: Purchase1 [4]

  Purchase1 Purchase2     n
     <fctr>    <fctr> <int>
1 Product_A Product_A   548
2 Product_D             548
3 Product_B             346
4 Product_C Product_C   345
5 Product_B Product_B   291
6 Product_D Product_D   281
```

可以看出，最频繁的序列是购买一次ProductA后再购买一次ProductA，以及购买一次ProductD后不再购买。有趣的是同样商品购买之间的频率。

下面使用TraMineR包开始更深入的检查。要使用TraMineR包，应该先转换数据，使用seqdef()函数将数据保存到一个序列类的对象中。这个对象只包含序列数据，没有任何协变量。还有，可以使用参数xtstep = n指定绘图函数中刻度线的距离。在这个例子中，每个事件使用1个刻度线：

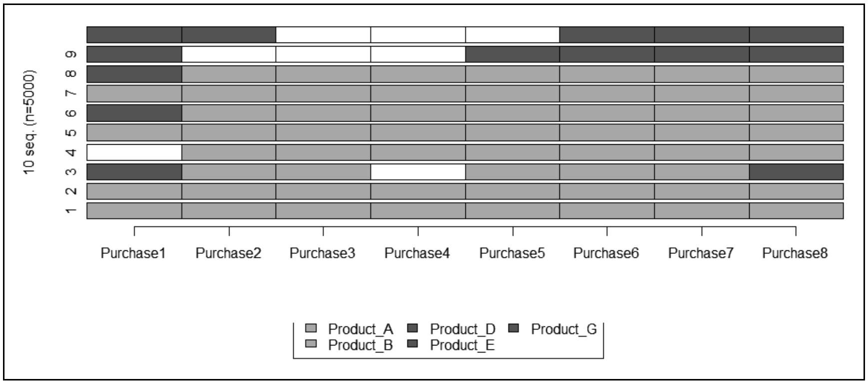
```
> seq <- seqdef(df[, -1], xtstep = 1)

> head(seq)
  Sequence
1 Product_A-Product_A------
2 Product_B-------
3 Product_G-Product_B-Product_B-Product_C-Product_B-Product_B-
Product_B-
  Product_G
4 Product_C-------
5 Product_A-------
6 Product_D-------
```

下面进行更深入的数据探索，查看索引图，可以生成前10个观测的序列。你可以使用数据索引检查任意多的观测和事件区间：

```
> seqiplot(seq)
```

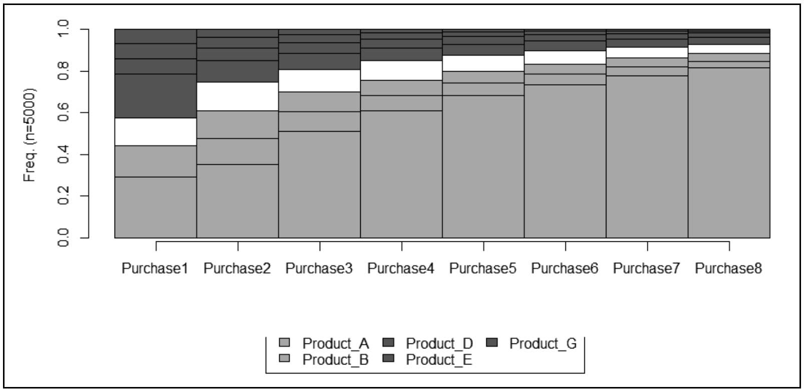
上述命令输出如下。



使用seqIplot()函数绘制所有观测，但因为数据量的原因，这样做没有什么意义。绘制状态分布图更能说明问题：

```
> seqdplot(seq)
```

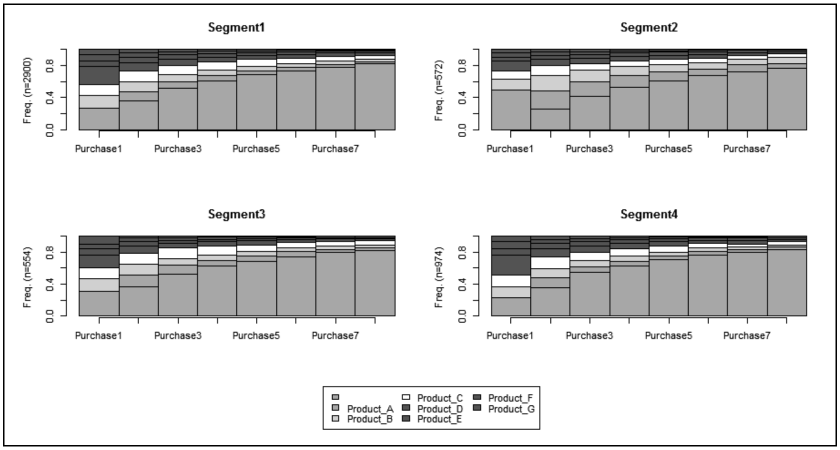
上述代码输出如下。



在这幅图中，可以明显看出商品购买按照状态的分布。还可以按照顾客类别对这幅图进行分组，看看不同顾客类别之间是否有区别：

```
> seqdplot(seq, group = df$Cust_Segment)
```

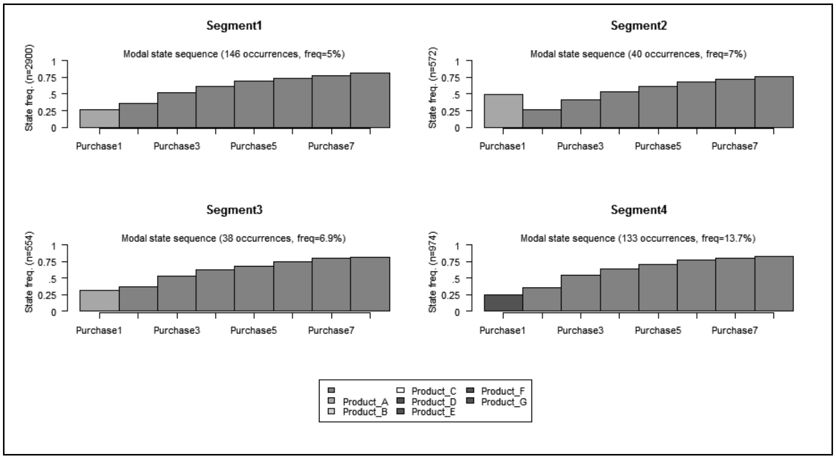
上述代码输出如下页图。



从上图中可以清楚地看出，Segment2与其他顾客分类相比，购买ProductA的比例是最高的。还可以通过模态图发现问题：

```
> seqmsplot(seq, group = df$Cust_Segment)
```

上述命令输出如下。



这个图很有意思。大约有50%的Segment2类别的顾客首次购买商品是ProductA，而对于Segment4类别的顾客，最频繁的首次购买商品则是ProductD。另一种可能有意义的图是时间均值图，但我认为这个例子不需要它。时间均值图可以绘制出每种状态下花费的平均“时间”。因为这个例子不是基于时间的，所以它没有意义，但我认为还是应该提一句：

```
> seqmtplot(seq, group = df$Cust_Segment)
```

继续前面的代码，深入查看序列变化。下面的代码先创建一个序列对象，筛选出那些出现频

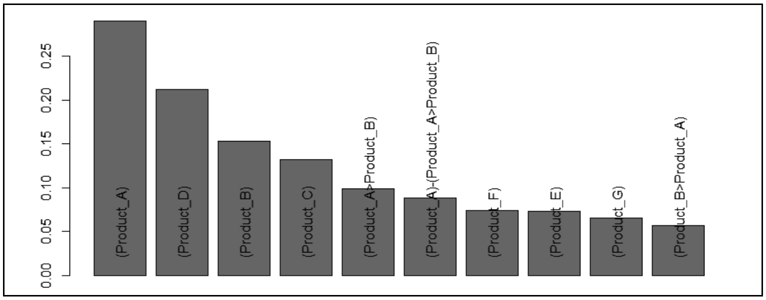
率大于5%的序列，然后绘制前10个序列：

```
> seqE <- seqecreate(seq)

> subSeq <- seqefsub(seqE, pMinSupport = 0.05)

> plot(subSeq[1:10], col = "dodgerblue")
```

上述命令输出如下。



请注意，这张图表示出序列在8种转换状态下的频率百分比。如果想进行简化，比如只使用前两种转换，可以在seqecreate()函数中设定索引。

最后，看看如何使用数据创建一个转换矩阵，从而表示从一个状态转换到另一个状态的概率。正如前面所说，这个矩阵还可以用于马尔科夫链的模拟，对未来进行预测。这已经超出了本章的范围，如果你感兴趣，我建议你研究R中的markovchain软件包及其教程，看看如何实现这个过程。我们给出了两种可能的转换矩阵。

一种矩阵包括所有状态之间的全部概率，另一种矩阵则表示从某个状态转换到另一种状态的概率，也就是时间变化矩阵。要想建立第二种矩阵，只要在函数中设定参数time.varying=TRUE即可。下面的代码展示了如何建立第一种矩阵：

10

```
> seqMat <- seqtrate(seq)
 [>] computing transition rates for states

 /Product_A/Product_B/Product_C/Product_D/
   Product_E/Product_F/Product_G ...

> options(digits = 2) # make output easier to read

> seqMat[2:4, 1:3]
                 [-> ] [-> Product_A] [-> Product_B]
[Product_A ->]  0.19          0.417          0.166
[Product_B ->]  0.26          0.113          0.475
[Product_C ->]  0.19          0.058          0.041
```

上面的输出显示了矩阵中的第2~4行和第1~3列。从矩阵中可以看出，购买ProductA之后，有

差不多42%的概率还会再购买一次ProductA，有19%的概率不再购买商品，有17%的概率会购买ProductB。我们要检查的最后一项结果是对于每种购买行为，随后不再购买商品的概率：

```
> seqMat[, 1] [ ->] [Product_A ->] [Product_B ->] [Product_C ->]
  [Product_D ->]
 1.00           0.19           0.26           0.19           0.33
[Product_E ->] [Product_F ->] [Product_G ->]
          0.18           0.25           0.41
```

从矩阵中可以看出，如果没有购买行为，那么随后也不再购买商品的概率为100%，这是很自然的事情。还有，我们发现，获得ProductD之后，不再购买商品的概率是33%。这是Segment4类别的顾客吗？有可能。

这个分析只需几行代码即可完成，而且不需要使用Excel或者其他昂贵的可视化软件，真的太棒了。你有纵向数据吗？如果有，那就试试序列分析吧！

10.10　小结

我们在本章的目标是介绍如何使用R进行关联规则挖掘（购物篮分析），以及建立并测试推荐引擎。购物篮分析就是试图理解可能同时购买哪些商品。推荐引擎的目标就是，在顾客对以前浏览过和购买过的商品进行评价的基础上，向其推荐感兴趣的其他商品。需要注意的一点是，介绍推荐系统时使用的R包（recommederlab）不是为了实现推荐算法而开发的，而是基于算法研究与测试的目的。本章介绍的另外一项内容是如何对纵向数据进行挖掘，以获得有用的知识，在我们的例子中就是顾客购买商品的顺序。这种分析方法有很多实际用途，包括市场营销活动以及卫生保健。

下面我们再回到监督式学习。第11章将介绍一些最激动人心的重要方法以进行机器学习实战，那就是多类分类和建立集成模型。如果使用近期发布的一些软件包，那么用R完成这些任务会非常容易。

第 11 章

创建集成多类分类

“赢得机器学习竞赛的手段就是：把别人的工作拿过来，再揉到一起。”

——维塔利·库兹涅佐夫，2014年神经信息处理系统大会

你可能已经意识到，我们讨论过集成学习。www.scholarpedia.org对集成学习的定义是：“有策略地建立多个模型（如分类器或专家系统）并将其组合在一起，解决特定计算智能问题的过程。”在随机森林和梯度提升模型中，我们将几百或几千棵树的“投票”结果组合起来进行预测。于是，根据集成学习的定义，这些模型就是集成学习模型。这种方法可以扩展至任意学习器，以建立集成模型，有人将这样的学习器称为元组件或元学习器。我们要介绍一种名为“模型融合”的方法。在这种方法中，我们要生成多个分类器，然后将这些分类器的类别预测概率作为另一个分类器的输入特征。这种方法可以提高预测准确率。前面的章节重点讨论了二值结果的分类问题，本章将讨论数据包含的结果多于两个时的预测方法。在真实世界的数据集中，这种情况非常常见。我必须承认，R语言中这些方法的应用才是我遇到过的最有趣、最令人享受的过程。

11.1 集成模型

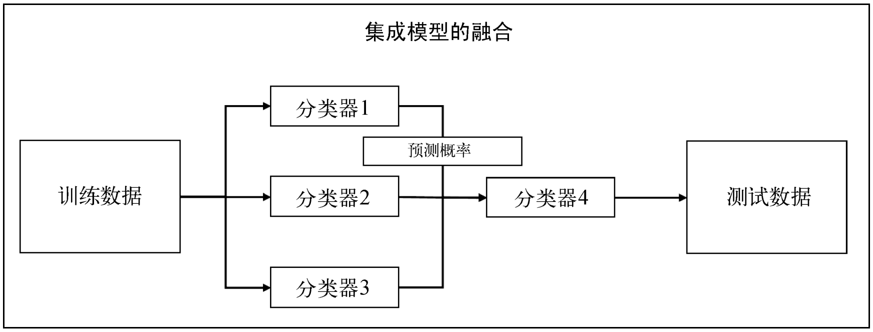
本章开头的引文提到，可以使用集成模型赢得机器学习竞赛。实际上，集成模型确实有实际应用。我已经提供了一个集成模型的定义，但它为什么有效？为了说明这个问题，我从下面的博客选择了一个例子，其中深入讲述了一些集成学习的方法：

http://mlwave.com/kaggle-ensembling-guide/

我撰写本章时，距离第51届超级碗比赛只有几天时间，当时亚特拉大猎鹰队对阵新英格兰爱国者队。假设我们在熟人之间下了一笔小小的赌注，赌爱国者队净负3分，要评估一下赢得这笔赌注的概率。再假设我们跟随3位专业预言家的预测，他们给出的爱国者队净负3分以上的概率都是60%。所以，如果我们相信任何一个所谓专家的预测，那么很明显，有60%的概率可以赢得赌注。但是，对专家的预测建立一个集成模型可以提高我们获胜的概率，这个概率使我们既能赢得亲朋好友的钱，又可以无情地嘲笑他们。下面看看如何做到这一点。

首先计算专家选择新英格兰队的每种可能结果的概率。如果3位专家都选择了新英格兰队，那么他们都正确的概率是$0.6 \times 0.6 \times 0.6$，即21.6%。如果任意两位选择了新英格兰队，那么有$(0.6 \times 0.6 \times 0.4) \times 3$，即43.2%的概率获胜。通过少数服从多数的原则，如果3人中至少2人选择了新英格兰队，那么我们获胜的概率差不多是65%。

这个例子相当简单，但非常有代表性。在机器学习中，这种方法的优点是可以将几种性能平平甚至很差的学习器的预测结果结合起来，从而提高整体准确率。下图说明了这个过程是如何完成的。



在这幅图中，我们建立了3种不同的分类器，并使用它们的预测概率作为第4个不同分类器的输入，以对测试数据做出预测。下面看看如何使用R语言实现。

11.2　业务理解与数据理解

本节会再一次遇到老冤家：皮玛印第安人糖尿病数据集。20世纪70年代中期，对于那些以正确率衡量性能的分类器来说，这个数据集就已经被证明是个超级难题。第5章和第6章已经研究过这个数据集，所以就不介绍它的细节了。有很多R包可以建立集成模型，即使你自己用代码实现也不难。本节使用caret包和caretEnsemble包解决这个问题。先加载程序包并进行数据准备，使用caret包中的`createDataPartition()`函数创建训练集和测试集：

```
> library(MASS)

> library(caretEnsemble)

> library(caTools)

> pima <- rbind(Pima.tr, Pima.te)

> set.seed(502)

> split <- createDataPartition(y = pima$type, p = 0.75, list = F)

> train <- pima[split, ]

> test <- pima[-split, ]
```

11.3 模型评价与模型选择

和我们在前面的章节中所做的一样，使用caret包的功能时，第一件事就是建立一个对象，以设定如何训练模型。可以使用`trainControl()`函数完成。要通过5折交叉验证来建立模型，并将最后预测（概率）保存下来。你还应该按照索引进行再抽样，以使每个基础模型都使用同样的数据折进行训练。此外，还要注意我在函数中设定了上采样。为什么呢？因为Yes和No的比例是2：1：

```
> table(train$type)

  No Yes
267  133
```

这个比例并不一定意味着分类不平衡，但我想在这里说明一些问题。在很多数据集中，我们感兴趣的结果是一种罕见的事件。于是你就会得到这样一种分类器，它的正确率非常高，但预测我们感兴趣的结果时效果非常差。也就是说，根本预测不出真阳性的结果。为了平衡响应变量，你可以对少数类进行上采样，对多数类进行下采样，建立"人工合成"的数据。下一个练习会重点讨论合成数据，在这个练习中，我们试试上采样。在上采样过程中，对于交叉验证使用的每折数据，少数类都使用有放回的随机抽样，以使其数量与多数类的观测数量相匹配。我们的函数形式如下：

```
> control <- trainControl(method = "cv",
  number = 5,
  savePredictions = "final",
  classProbs = T,
  index=createResample(train$type, 5),
  sampling = "up",
  summaryFunction = twoClassSummary)
```

然后使用`caretList()`函数训练模型。在函数中，你可以使用任意caret包支持的模型，可用的模型列表及其相应的超参数参见：

https://rdrr.io/cran/caret/man/models.html

在这个例子中，我们训练3种模型。

- 分类树模型：`"rpart"`
- 多元自适应回归样条模型：`"earth"`
- K最近邻模型：`"knn"`

可以同时训练这3种模型：

```
> set.seed(2)

> models <- caretList(
```

```
  type ~ ., data = train,
  trControl = control,
  metric = "ROC",
  methodList = c("rpart", "earth", "knn")
  )
```

caretList()函数不但能够建立模型，还可以按照caret包的规则调整每种模型的超参数。使用caretModelSpec()函数可以为每种模型创建各自的调优参数网格，但出于演示的目的，我们将调优工作交给caretList()函数。可以调用模型对象检查结果。以下是一些简化的输出：

```
> models
...
Resampling results across tuning parameters:

        cp       ROC      Sens      Spec
0.03007519 0.7882347 0.8190343 0.6781714
0.04010025 0.7814718 0.7935024 0.6888857
0.36090226 0.7360166 0.8646440 0.6073893
```

基础模型能够有效组合的要求是，它们不高度相关。这是一个主观的判定，模型预测结果是否相关没有严格的规则。你应该用结果进行实验，必要时可以替换一个模型。看看我们的结果：

```
> modelCor(resamples(models))
          rpart     earth       knn
rpart 1.0000000 0.9589931 0.7191618
earth 0.9589931 1.0000000 0.8834022
knn   0.7191618 0.8834022 1.0000000
```

分类树与earth模型高度相关。这可能是一个问题，但我们先忽略它，继续建立第四个分类器——融合模型——并检查结果。要完成这一步，需要找出测试集中Yes的预测概率，并保存在数据框中：

```
> model_preds <- lapply(models, predict, newdata=test, type="prob")

> model_preds <- lapply(model_preds, function(x) x[,"Yes"])

> model_preds <- data.frame(model_preds)
```

下面使用caretStack()函数将这些模型融合在一起，进行最终预测。这就是基于5个自助抽样样本的一个简单逻辑斯蒂回归：

```
> stack <- caretStack(models, method = "glm",
  metric = "ROC",
  trControl = trainControl(
  method = "boot",
  number = 5,
  savePredictions = "final",
  classProbs = TRUE,
  summaryFunction = twoClassSummary
  ))
```

检查最终模型，如下所示：

```
> summary(stack)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1029  -0.6268  -0.3584   0.5926   2.3714

Coefficients:
            Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)   2.2212      0.2120   10.476  < 2e-16 ***
rpart        -0.8529      0.3947   -2.161  0.03071 *
earth        -3.0984      0.4250   -7.290  3.1e-13 ***
knn          -1.2626      0.3524   -3.583  0.00034 ***
```

即使rpart模型和earth模型高度相关，它们的系数也依然是显著的，所以应该可以在分析中保留这两个模型。下面比较单独模型的结果和集成模型的结果：

```
> prob <- 1-predict(stack, newdata = test, type = "prob")

> model_preds$ensemble <- prob

> colAUC(model_preds, test$type)
               rpart     earth       knn  ensemble
No vs. Yes 0.7413481 0.7892562 0.7652376 0.8001033
```

通过`colAUC()`函数可以看到单独模型的AUC和集成模型的AUC。集成模型的结果要比只使用earth包中的多元自适应回归样条模型的结果稍好一点。所以由本例可以发现，通过模型融合建立的集成模型确实可以提高预测能力。对于这个数据集，你能建立一个更好的集成模型吗？你应该使用什么样的抽样方法或分类器呢？带着这些问题，我们开始研究多类分类。

11.4 多类分类

有多种方法可以学习多类分类问题。随机森林和判别分析这样的技术可以处理多类问题，但有些技术和软件包无能为力，比如基础R包中的广义线性模型`glm()`。不幸的是，上面提到的caretEnsemble包无法处理多类分类问题，但R语言机器学习包（mlr）可以同时支持多类分类和集成方法。如果你非常熟悉Python中的scikit-Learn扩展包，那就没什么问题，因为mlr就在R语言中提供了与之相同的功能。mlr和基于caret的软件包迅速成为我解决几乎所有商业问题的首选。我本节要演示mlr包解决多类分类问题时有多么强大，最后还要展示如何在皮玛印第安人数据集上建立集成模型。

对于多类分类问题，我们先看看如何调整随机森林模型，然后再研究如何使用“一对多”技术使一个GLM模型转换为多类分类学习器。这种技术先将某个类别归为一类，其他类别归为另

一类，建立一个二值概率预测模型，对每个类别都做完上述操作之后，再将结果组合起来预测观测的最终类别。这种技术可以使你将任何分类器扩展到多类分类问题上，其效果经常会超过多类分类学习器。

提醒一下，不要混淆多类分类和多标签这两个术语。对于多类分类，一个观测可以被分到而且只能被分到一个类别中；而对于多标签，观测可以被分配到多个类别中。多标签的一个例子是文本既可以被标记为“政治”，也可以被标记为“幽默”。本章不会涉及多标签问题。

11.5　业务理解与数据理解

我们继续使用第8章用过的葡萄酒数据集。它包括13个数值型特征和1个响应变量，响应变量中有3个普通酒类别。我们的任务就是预测这些类别。我要对数据集进行有趣的修改，人工增加观测数量。原因有两点，第一，我要充分展示mlr包的再抽样能力；第二，我要介绍一种合成抽样的技术。上一节介绍了上采样，这次要介绍合成数据的技术。

第一项任务是加载程序包及数据：

```
> library(mlr)

> library(ggplot2)

> library(HDclassif)

> library(DMwR)

> library(reshape2)

> library(corrplot)

> data(wine)

> table(wine$class)

 1  2  3
59 71 48
```

我们有178条观测，响应变量是数值型的标签（1、2和3）。下面使数据量倍增。这个例子中使用的算法称为**合成少数类过采样技术**。前一个例子使用了上采样，对少数类进行了**有放回的采样**，直至其数量可以和多数类相匹配。通过SMOTE，我们先对少数类进行随机抽样，再计算（识别）出每个观测的K最近邻，然后基于这些近邻随机生成数据。在DMwR包中的`SMOTE()`函数中，默认的最近邻数量为5。另一件需要考虑的事情是少数类过采样的比例。举例来说，如果想创建数量是现有数量2倍的少数类，就应该在函数中设定`"percent.over = 100"`。对于要添加到现有少数类中的每个观测，新采样的数量为percent.over的值除以100，即每个观测有1个新采样。还有另外一个采样比例参数，它控制着在新数据集中对多数类进行随机选择的数量。

以下就是这项技术的实际应用代码，首先将类别转换为因子变量，否则函数不会起作用：

```
> wine$class <- as.factor(wine$class)

> set.seed(11)

> df <- SMOTE(class ~ ., wine, perc.over = 300, perc.under = 300)

> table(df$class)

  1   2   3
195 237 192
```

瞧！我们创建了一个具有624个观测的数据集。下一项任务是按类别对各种特征进行可视化。我是箱线图的狂热粉丝，所以先对前4个输入特征按类别做出箱线图。这4个特征的量度都不一样，将它们放入一个均值为0、标准差为1的数据框中将有助于比较：

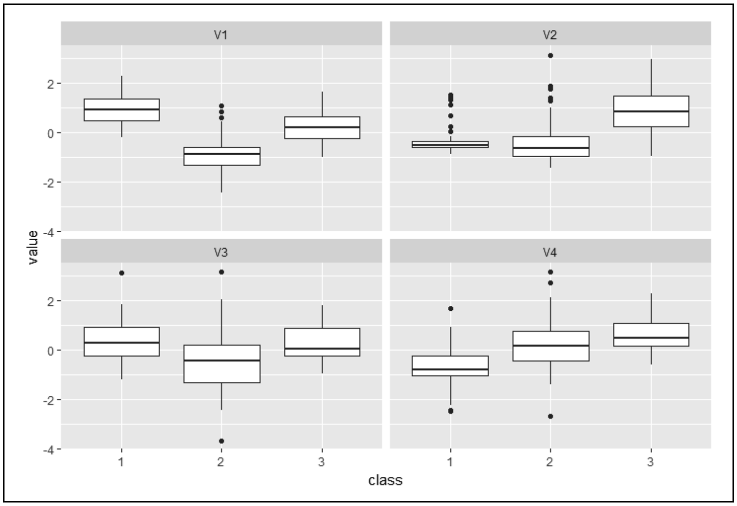
```
> wine.scale <- data.frame(scale(wine[, 2:5]))

> wine.scale$class <- wine$class

> wine.melt <- melt(wine.scale, id.var="class")

> ggplot(data = wine.melt, aes( x = class, y = value)) +
  geom_boxplot() +
  facet_wrap( ~ variable, ncol = 2)
```

上述命令输出如下。



回忆一下第3章，箱线图中的圆点被认为是离群点。那么怎么处理离群点呢？有以下几种方式：

- ❑ 什么也不做——这总是一种选择；

- ❑ 删除这些对应于离群点的观测；
- ❑ 在当前特征内将观测截断，或者为截断的数据新建一个特征值；
- ❑ 为每个特征创建一个因子变量，以确定一个观测是否是离群点。

我总觉得离群点很有意思，也经常对它们进行仔细的研究，以确定为什么会出现离群点以及如何处理离群点。这个例子时间不够，所以简单处理，对离群点进行分段编码。创建一个函数，识别离群点。对于其中的高异常值（高于第99个百分位），将其设为第75个百分位的值；对于其中的低异常值（低于第1个百分位），将其设为第25个百分位的值。你也可以将这些异常值设为中位数，但我发现上面的处理方式效果很好。

> 你可以将这些代码段放在同一个函数中，但我认为分成两部分更简单，也更容易理解。

下面就是处理离群点的函数：

```
> outHigh <- function(x) {
  x[x > quantile(x, 0.99)] <- quantile(x, 0.75)
  x
  }

> outLow <- function(x) {
  x[x < quantile(x, 0.01)] <- quantile(x, 0.25)
  x
  }
```

现在在初始数据上运行函数，创建新数据框：

```
> wine.trunc <- data.frame(lapply(wine[, -1], outHigh))

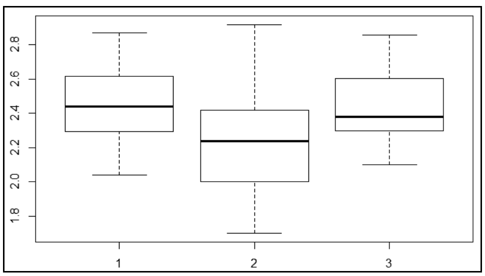
> wine.trunc <- data.frame(lapply(wine.trunc, outLow))

> wine.trunc$class <- wine$class
```

可以将截断后的数据与初始数据进行简单的对比。用V3这个特征试试：

```
> boxplot(wine.trunc$V3 ~ wine.trunc$class)
```

上述命令输出如下。

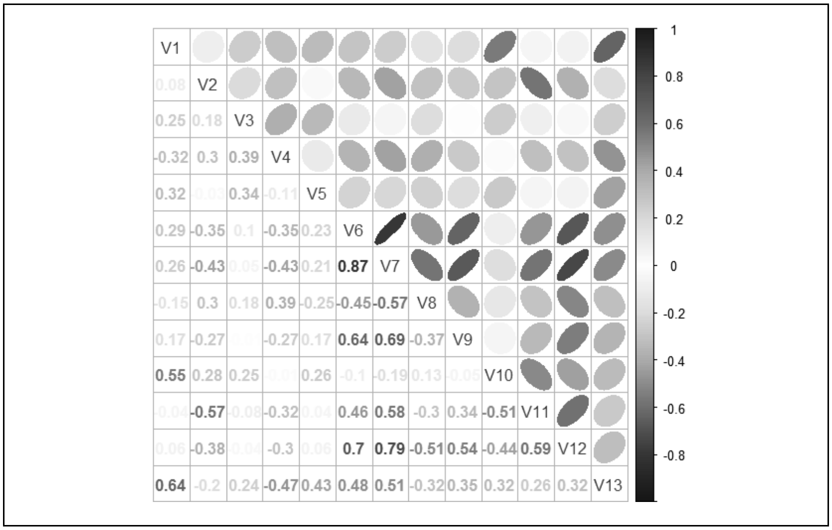


可以看出，效果非常好。下面看看相关性：

```
> c <- cor(wine.trunc[, -14])

> corrplot.mixed(c, upper = "ellipse")
```

上述命令输出如下。



可以看出，V6和V7高度相关，有一个高于0.5的值。一般来说，在基于非线性的学习方法中，这不是一个问题。但在我们的GLM模型中，要加入一个L2惩罚项（岭回归）来解决它。

11.6 模型评价与模型选择

从创建训练集和测试集开始，然后创建一个随机森林分类器作为基础模型。对这个模型的表现进行评价之后，继续实验一对多类分类方法，看看其效果如何。我们按照70/30的比例划分数据。此外，mlr包的一个特点是，它要求将训练集数据保存在task数据结构中，该数据结构专门为分类任务而设计。也可以将测试集数据放在task数据结构中，但这不是必须的。

完整的模型列表参见如下链接，当然，你可以使用自己的模型：

https://mlr-org.github.io/mlr-tutorial/release/html/integrated_leaners/index.html

```
> library(caret) #if not already loaded

> set.seed(502)

> split <- createDataPartition(y = df$class, p = 0.7, list = F)
```

```
> train <- df[split, ]

> test <- df[-split, ]

> wine.task <- makeClassifTask(id = "wine", data = train, target =
  "class")
```

11.6.1　随机森林

创建训练集数据的task对象之后，可以用多种函数探索数据。下面是结构函数的简化输出：

```
> str(getTaskData(wine.task))
'data.frame': 438 obs. of 14 variables:
 $ class: Factor w/ 3 levels "1","2","3": 1 2 1 2 2 1 2 1 1 2 ...
 $ V1 : num 13.6 11.8 14.4 11.8 13.1 ...
```

在分析中，有多种方式使用mlr，但我建议你先创建一个重抽样对象。在本例中，我们创建一个重抽样对象来为随机森林模型调整树的数量，它包括3个子样本：

```
> rdesc <- makeResampleDesc("Subsample", iters = 3)
```

下一个对象建立了一个树的网格来调整树的数量，最小值为750，最大值为2000。你还可以像我们使用caret包时那样，建立多个参数列表。查看`makeParamSet`函数的帮助文档，研究其设置选项：

```
> param <- makeParamSet(
    makeDiscreteParam("ntree", values = c(750, 1000, 1250, 1500,
     1750, 2000))
)
```

下面创建一个控制对象，建立数值网格：

```
> ctrl <- makeTuneControlGrid()
```

这样即可调整超参数，从而找出最优树数量。然后，调出最优树数量和相应的样本外误差：

```
> tuning <- tuneParams("classif.randomForest", task = wine.task,
    resampling = rdesc, par.set = param,
    control = ctrl)

> tuning$x
$ntree
[1] 1250

> tuning$y
mmce.test.mean
0.01141553
```

最优的树数量是1250，相应的平均误分类率是0.01%，几乎是一个完美的分类。于是，使用这个超参数作为`makeLearner()`函数的训练包装器就非常简单了。请注意，我将预测类型设置

为概率，因为默认类型是预测分类：

```
> rf <- setHyperPars(makeLearner("classif.randomForest",
  predict.type = "prob"), par.vals = tuning$x)
```

下面训练模型：

```
> fitRF <- train(rf, wine.task)
```

可以看看训练数据上的混淆矩阵：

```
> fitRF$learner.model
          OOB estimate of error rate: 0%
Confusion matrix:
   1   2   3  class.error
1 72   0   0            0
2  0  97   0            0
3  0   0 101            0
```

然后在测试集上评价模型效果，看看误差率和正确率（1 – 误差率）。因为没有测试数据的task对象，所以需要指定newdata = test。如果你建立了测试数据的task对象，那么使用test.task即可：

```
> predRF <- predict(fitRF, newdata = test)

> getConfMatrix(predRF)
       predicted
true      1  2  3  -SUM-
  1      58  0  0      0
  2       0 71  0      0
  3       0  0 57      0
  -SUM-   0  0  0      0

> performance(predRF, measures = list(mmce, acc))
 mmce acc
  0   1
```

我们很轻松地预测了每个类别，没有一点错误。

11.6.2 岭回归

出于演示的目的，我们还是要使用一对多的方法实验岭回归。要想完成这个任务，需要为二值分类方法创建一个MulticlassWrapper对象。classif.penalized.ridge方法来自penalized软件包，所以请先确认你已经安装了它：

```
> ovr <- makeMulticlassWrapper("classif.penalized.ridge",
  mcw.method = "onevsrest")
```

下面为分类器创建一个包装器，包装器可以用装袋法进行10次（默认值）有放回重抽样，抽取70%的观测和所有输入特征：

```
> bag.ovr = makeBaggingWrapper(ovr, bw.iters = 10, #default of 10
  bw.replace = TRUE, #default
  bw.size = 0.7,
  bw.feats = 1)
```

现在使用这个对象训练算法。请注意，我在代码中的`train()`函数前面加上了`mlr::`，因为caret包中也有`train()`函数，所以要设定使用mlr包中的函数，而不是caret包中的。如果同时加载了两个包，不如前设置就会导致严重的错误：

```
> set.seed(317)
> fitOVR <- mlr::train(bag.ovr, wine.task)
> predOVR <- predict(fitOVR, newdata = test)
```

结果如下所示：

```
> head(data.frame(predOVR))
  truth response
60     2        2
78     2        2
79     2        2
49     1        1
19     1        1
69     2        2

> getConfMatrix(predOVR)
       predicted
true      1  2  3  -SUM-
  1      58  0  0      0
  2       0 71  0      0
  3       0  0 57      0
  -SUM-   0  0  0      0
```

小菜一碟。但别太注重正确率，应该更注重创建分类器、调整超参数和实现重抽样的策略和方法。

11.7　MLR 集成模型

我们还有一块硬骨头：皮玛印第安人糖尿病数据的分类。和使用caret包一样，你也可以用mlr包建立集成模型。我也会展示如何在学习过程中加入SMOTE技术，而不只创建单独的数据集。

首先，确认你运行了本章开始时的代码，创建训练集和测试集。我在这里暂停一下，以便你完成这个操作。

好的，和前面一样，创建训练集的task对象：

```
> pima.task <- makeClassifTask(id = "pima", data = train, target =
  "type")
```

此处的smote()函数和前面的用法有些不一样。你只需设定少数类过采样的比率和K最近邻的数量。我们基于3个最近邻来使少数类（Yes）加倍：

```
> pima.smote <- smote(pima.task, rate = 2, nn = 3)

> str(getTaskData(pima.smote))
'data.frame': 533 obs. of 8 variables:
```

现在训练集中有533个观测，而不是初始的400个。为了完成集成模型的融合，我们要建立3个基础模型（随机森林、二次判别分析和带有L1惩罚项的GLM）。以下代码将它们组合在一起作为基础模型，如果你愿意，也可以加入一个学习器，并确保使用基础模型的预测概率作为输入特征：

```
> base <- c("classif.randomForest", "classif.qda", classif.glmnet")

> learns <- lapply(base, makeLearner)

> learns <- lapply(learns, setPredictType, "prob")
```

融合模型是一个简单的GLM模型，它的系数是通过交叉验证调整得来的。软件包的默认设置是使用5折交叉验证：

```
> sl <- makeStackedLearner(base.learners = learns,
  super.learner = "classif.logreg",
  predict.type = "prob",
  method = "stack.cv")
```

这样即可训练基础模型和融合模型。如果你觉得合适，可以选择加入重抽样和调优包装器，就像我们在前一节做的那样。这个例子使用默认设置即可。测试集上的训练和预测也是一样的：

```
> slFit <- mlr::train(sl, pima.smote)

> predFit <- predict(slFit, newdata = test)

> getConfMatrix(predFit)
      predicted
true        No Yes -SUM-
 No        70  18    18
 Yes       15  29    15
 -SUM-     15  18    33

> performance(predFit, measures = list(mmce, acc, auc))
  mmce    acc        auc
  0.25   0.75   0.7874483
```

通过这些努力，我们只得到了75%的正确率，AUC与用caretEnsemble包建立的集成模型相比，还差了那么一点点，因为我们使用了不同的基础学习器。所以，又回到前面的问题了：你能改善这个结果吗？请让我知道你的努力与成果。

11

11.8　小结

本章先介绍了通过融合方法建立集成模型这一非常重要的机器学习方法，然后讨论了多类分类问题。在融合方法中，我们使用基础模型（学习器）预测概率，并使用这些概率作为另一个模型（超级学习器）的输入特征来做出最终预测。融合模型与单独的基础模型相比，确实在性能上有所提高。对于多类分类方法，我们既介绍了其使用方法，也讲解了如何通过一对多技术将二值分类方法应用于多类分类问题。我们还顺便介绍了两种可以平衡类别的抽样技术（上采样与合成少数类过采样技术）。此外的重要内容就是两种功能强大的R包caretEnsemble和mlr的使用方法，这些方法和程序包对于R机器学习实践者来说都是非常强大的辅助工具。

下一章将进入时间序列与因果关系的世界。在我看来，时间序列分析是最容易被误解和忽视的机器学习领域之一。学完下一章，你将可以帮助我们的同行尽快消除这一误解。

第 12 章 时间序列与因果关系

“经济学家总是要到明天才能知道为什么昨天预言的事情没有在今天发生。”

——劳伦斯·彼得

单变量时间序列就是按照标准的时间间隔进行收集的测量结果，时间间隔可以是分钟、小时、天、周或者月。时间序列与使用其他方式收集的数据相比具有特殊的问题，因为观测顺序非常重要。这种对观测顺序的依赖会使标准分析方法产生不必要的偏差和方差。

在机器学习中，时间序列数据看上去非常枯燥无味。不幸的是，太多实际数据和时间相关，而且时间序列分析可能会非常复杂和棘手。可以说，如果你没有见过糟糕的时间序列分析，那肯定是因为你没有仔细研究过它。

另一个存在于时间序列中但经常被忽视的问题是因果关系。是的，我们没有将相关性和因果关系混为一谈，但是在时间序列分析中，可以使用格兰杰因果关系模型这样的技术来确定是否存在统计角度的因果关系。

本章要应用时间序列（或计量经济学）技术确定单变量预测模型、向量自回归模型，以及最后的格兰杰因果关系模型。结束本章之后，你可能算不上时间序列分析大师，但你的知识已经足以进行有效分析，并且可以明确一些基本问题，这些问题都是你在建立时间序列模型和预测模型时要考虑的。

12.1 单变量时间序列分析

我们着重讨论两种分析和预测单变量时间序列的方法：**指数平滑**和**自回归移动平均**模型。先从指数平滑方法开始。

与移动平均模型相似，指数平滑模型对过去的观测进行加权。但和移动平均模型不同的是，在指数平滑模型中，相对于更远期的观测来说，越是近期的观测，所得的权重越高。有3个可能的平滑参数需要估计：整体平滑参数、趋势参数和平滑参数。如果不存在趋势或季节性因素，相

应的参数就失效。

平滑参数使用下面的公式进行预测：

$$Y_t + 1 = \alpha(Y_t) + (1 - \alpha)Y_{t-1} + (1 - \alpha)^2 Y_{t-2} + \cdots, \text{当 } 0 < \alpha \leqslant 1$$

在这个公式中，Y_t是T时刻的值，α是平滑参数。算法通过最小化误差使α（和其他参数）达到最优，可以使用**误差平方和**，也可以使用**均方误差**。

预测公式以及趋势和季节性（如果可行）公式可以表示如下。

- 预测公式：A是前期平滑公式，h是预测阶段数，$Y_{t+h} = A + hB_t + S_t$
- 趋势公式：$B_t = \beta(A_t - A_{t-1}) + (1 - \beta)B_{t-1}$
- 季节性公式：m为季节性阶段的数量，$S_t = \Omega(Y_t - A_{t-1} - B_{t-1}) + (1 - \Omega)S_{t-m}$

这个预测公式被称为**Holt-Winters法**。预测公式本质上是可加的，趋势是线性的。这种方法也允许使用阻尼趋势和可乘季节性，此时的季节性随时间的变化成比例增加或减少。我的经验是，Holt-Winters法可以提供最好的预测，甚至比ARIMA模型还要好。我基于月度数据对几百个时间序列进行了长期预测才得到这个结论，在大概90%的案例中，Holt-Winter法产生的预测误差最小。此外，你也不用担心比如ARIMA模型中的平稳性假设。平稳性是指时间序列中所有阶段都具有固定的均值、方差和相关性。尽管如此，理解掌握ARIMA模型依然很重要，因为在某些情况下，这种模型表现最好。

在自回归模型中，时间T处的Y值是前一个Y值的线性函数。一阶自回归模型（AR(1)）的公式是$Y_t =$ 常数 $+ \Phi Y_{t-1} + E_t$。模型的关键假设如下：

- E_t是独立同分布的误差，均值为0，方差是一个常数；
- 误差与Y_t是独立的；
- Y_t、Y_{t-1}、$Y_{t-n}\cdots$是平稳的，这说明Φ的绝对值小于1。

在平稳的时间序列中，你可以查看**自相关函数**。平稳时间序列中的ACF可以给出Y_t和Y_{t-h}之间的相关性，其中$h = 1、2、\cdots、n$。我们使用R建立并绘制一个一阶自回归序列。绘图时还可以查看ggfortify包的功能，它可以作为ggplot2函数的包装器：

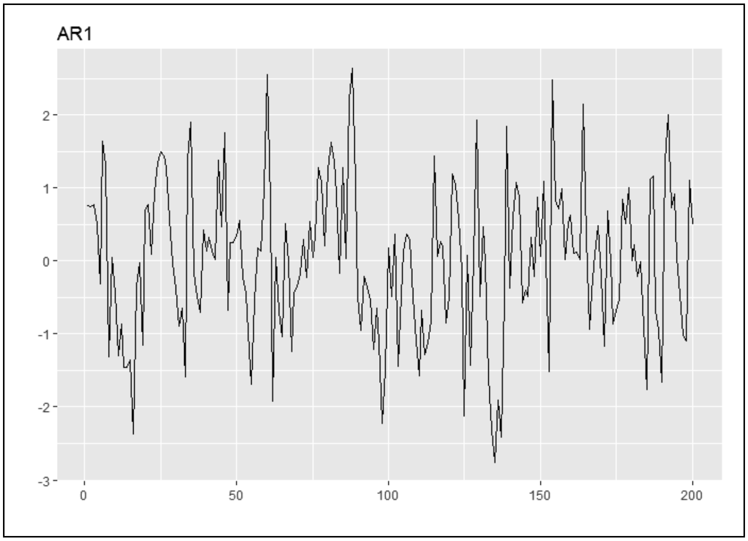
```
> library(ggfortify)

> set.seed(123)

> ar1 <- arima.sim(list(order = c(1, 0, 0), ar = 0.5), n = 200)

> autoplot(ar1, main = "AR1")
```

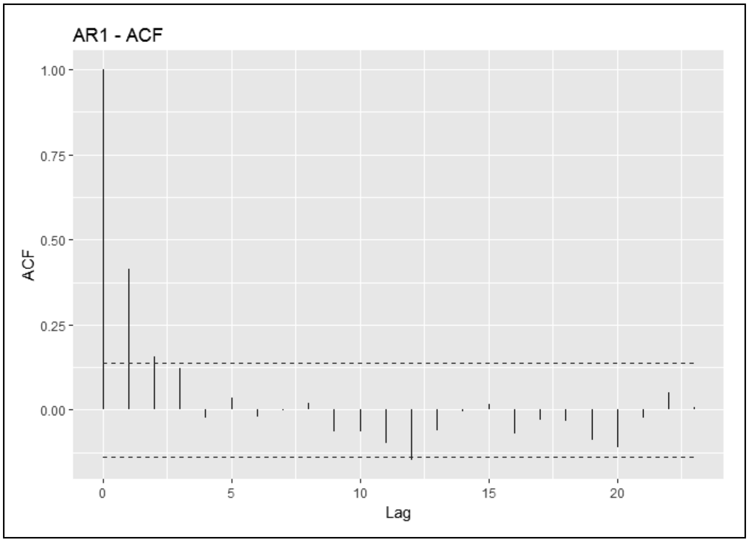
上述命令输出如下页图。



下面检查ACF：

```
> autoplot(acf(ar1, plot = F), main = "AR1 - ACF")
```

上述命令输出如下。

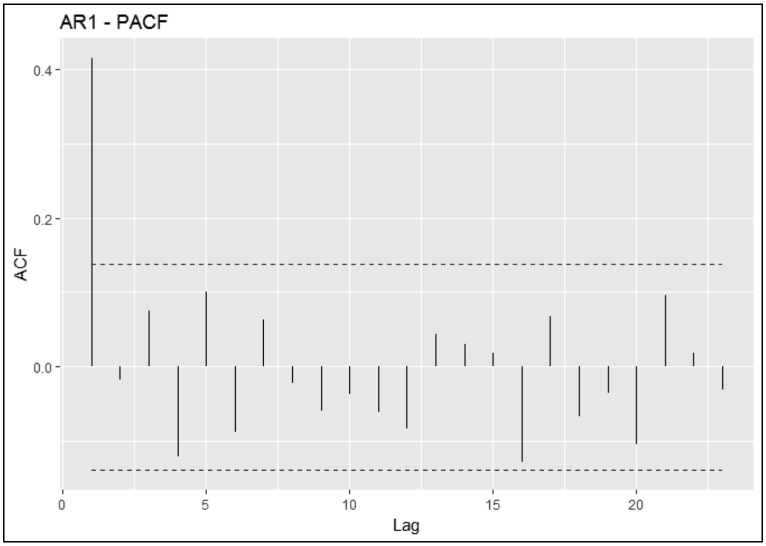


ACF图中显示，**延迟**（**Lag**）增加时，相关性呈指数减少。虚线表示显著相关性的置信带，任何一条超出置信带上界和下界的竖线都被认为是显著的相关性。除了ACF，还可以检查**偏自相关函数**。PACF计算的是条件相关性，表示Y_t与Y_{t-h}之间的相关性受二者之间的观测影响。对条件相关的一种直观理解是线性回归模型及其系数。假设有两个线性回归模型$Y = \beta_0 + \beta_1 X_1$和$Y = \beta_0 +$

$\beta_1X_1+\beta_2X_2$。在第一个模型中，X和Y之间的关系是线性的，可以用一个系数表示。但第二个模型中的情况就不一样了，因为必须考虑Y和X_2之间的关系。请注意下面的PACF图中，一阶偏自回归值和一阶自回归值是一样的，因为不存在条件相关性。

```
> autoplot(pacf(ar1, plot = F), main = "AR1 - PACF")
```

上述命令输出如下。



根据前面的时间序列图，我们完全可以假设序列是平稳的。在实际工作中，可以通过一些统计检验来确认数据是否平稳，但通常仅凭肉眼观察即可。如果数据不是平稳的，可能需要对数据进行差分来消除趋势。这就是ARIMA中Integrated（I）的意义。进行差分以后，新的序列为$\Delta Y_t=Y_t-Y_{t-1}$。我们希望通过一阶差分就可以得到稳定性，但某些情况下，二阶差分是必需的。具有一阶自回归（AR(1)）和一阶差分（I(1)）的ARIMA模型可以表示为ARIMA(1, 1, 0)。

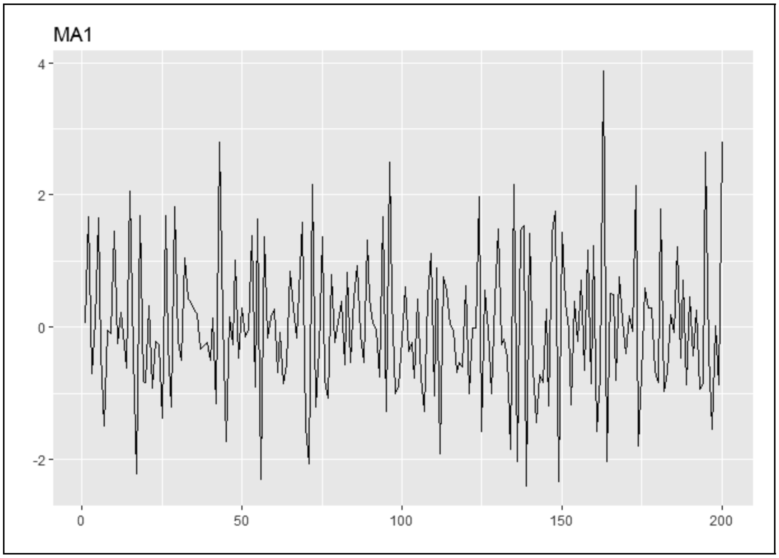
MA代表移动平均。与对股票价格进行50天移动平均不同，这不是一个简单的移动平均，它更像是应用在误差上的一个系数。当然，误差是独立同分布的，均值为0，方差是一个常数。一阶移动平均（MA(1)）的公式为$Y_t=$常数$+E_t+\Theta E_{t-1}$。和AR(1)模型一样，可以用R建立一个MA(1)模型，如下所示：

```
> set.seed(123)

> ma1 <- arima.sim(list(order = c(0, 0, 1), ma = -0.5), n = 200)

> autoplot(ma1, main = "MA1")
```

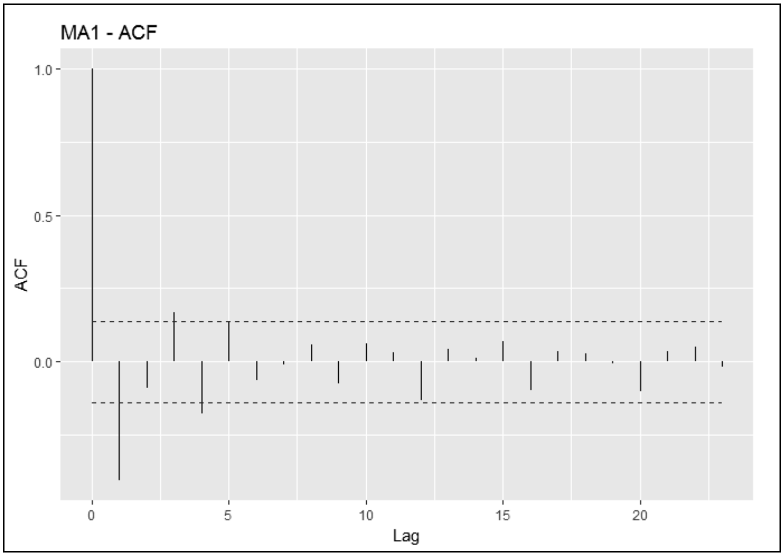
上述命令输出如下页图。



MA(1)的ACF和PACF图与AR(1)模型的有一点区别。请注意，查看图形时，有一条经验法则可以确定模型中是否有AR项和MA项。这可能稍嫌主观，所以你可以逐渐体会，但请相信，R对模型的判断是不会错的。通过下面的图可以看到一阶延迟具有显著的相关性，一阶延迟和二阶延迟中具有两种显著的偏相关性：

```
> autoplot(acf(ma1, plot = F), main = "MA1 - ACF")
```

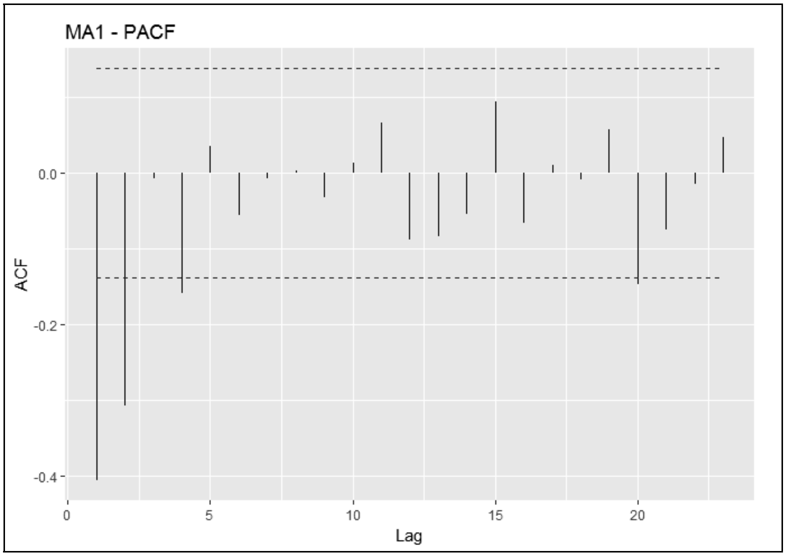
上述命令输出如下。



12

上页图是ACF图，下图是PACF图：

```
> autoplot(pacf(ma1, plot = F), main = "MA1 - PACF")
```



ARIMA模型包括自回归、差分和移动平均项，也可能包含季节性因素。非季节性ARIMA模型通常表示为(p, d, q)。在季节性ARIMA模型中，如果数据是月度的，那么可以表示为(p, d, q) × (P, D, Q)12，其中的12表示考虑了月度季节性。在我们将要使用的软件包中，R会自动识别是否应该包括季节性。如果应该，它会选择一个最优项。

理解格兰杰因果关系

假设有人问你："对于药X，新处方上的金额与全部处方的金额之间有什么联系？"你知道这些金额每月计算一次，所以如果人们确信新处方会增加总处方的金额，你应该如何理解它们之间的联系呢？或者，如何检验假设"大宗商品的价格——特别是铜——是美国股票市场价格的首要指标"？如果有两组时间序列数据x和y，格兰杰因果关系模型负责确定一个序列是否会影响另一个序列中的变化。它的做法是使用一个序列中的不同延迟序列为第二个序列中的变化建模。要达成这个目的，需要建立两个模型以预测y，一个模型只使用y的前期数据（Ω），另一个模型使用y的前期数据和x（π）。模型如下所示，其中k是时间序列中的延迟期数。

使 $\Omega = y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_k y_{t-k} + \in$

且使 $\pi = y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_k y_{t-k} + \alpha_1 y_{t-1} + \cdots + \alpha_k y_{t-k} + \in$

然后，比较RSS并使用F检验确定究竟是嵌套模型（Ω）能足够恰当地解释y的未来值，还是全模型（π）更好。F检验用来检验的原假设和备择假设如下所示。

- H0：对于所有$i \in [1, k]$，$\alpha_i = 0$，无格兰杰因果关系
- H1：至少存在一个$i \in [1, k]$，$\alpha_i \neq 0$，有格兰杰因果关系

本质上，我们试图确定能否在统计上证明，对于y的未来值，x可以比y的前期值提供更多信息。在这个定义中，很明显我们不是试图证明真正的因果性，而是两个值通过某种现象关联在一起。按照这个定义，还必须反向运行这个模型，以验证y没有对x的未来值提供信息。如果发现真的出现了这种情形，那就很可能存在一些外生变量（比如Z），或者需要我们控制，或者它们和y或x之间存在更强的格兰杰因果关系。为了避免虚假结果，应该在稳定的时间序列上进行处理。请注意，已经有一些研究性论文讨论了非线性模型使用的技术，但这已经超出了本书范围。不过我们会从非稳定序列的角度来研究它，有一篇精彩的介绍性论文是关于“鸡生蛋还是蛋生鸡”这一历史悠久的难题的（Thurman，1988）。

有几种不同的方式可以识别正确的延迟结构。当然，你可以使用暴力算法，不分青红皂白地把所有可能的延迟序列都测试一遍。基于专业知识或此前关于延迟选择指导的一些研究，你或许对这个问题具有一定的理性直觉。如果没有，就要使用**向量自回归**确定延迟结构，此时要使用最小信息准则，比如**赤池信息准则**或**最终预测误差**。为简单起见，下面给出了双变量VAR模型的表示方法，其中每个变量只包含一阶延迟。这种表示方法可以扩展为任意数量的变量和延迟：

- $\text{Y} = 常数_1 + \beta_{11}\text{Y}_{t-1} + \beta_{12}\text{X}_{t-1} + e_1$
- $\text{X} = 常数_2 + \beta_{21}\text{Y}_{t-1} + \beta_{22}\text{X}_{t-1} + e_2$

在R中可以非常简单地实现这个过程，如下面的实际案例所示。

12.2　业务理解

“地球永远都在那里。是我们！要灭绝的是我们自己。”

——乔治·卡林，哲学家、喜剧演员

气候正在发生变化，过去和未来一直如此，但重要（至少从政治和经济的角度看是重要的）的问题是，气候变化是人为的吗？本节将使用计量经济学时间序列模型进行实验，以确定碳排放量是否可以引起（统计意义上的）气候变化，更确切地说，使温度升高。我个人在这个问题上持中立态度，我永远也不会忘记卡林先生在关于这个问题的演讲中为我们留下的警示。

第一件事就是找到并收集数据。对于温度数据，应该选择HadCRUT4年度温度中位数时间序列，这可能是温度的黄金标准。这份数据由东英吉利大学气候研究所和英国气象局哈德莱研究中心合作编制，关于数据编制和模型化的完整介绍可以参考http://www.metoffice.gov.uk/hadobs/index.html。

我们将要使用的数据记录的是年度异常数据，它由某一年的年度地表温度中位数与参考年度（1961~1990）平均温度的差构成。年度地表温度是一个综合体，它由来自CRUTEM4地表空气温

度数据集和HadSST3海洋表面温度数据集的全球各地温度混合而成。这份数据最近遭到了质疑，人们认为它具有偏差，不那么可靠，详情参见http://www.telegraph.co.uk/comment/11561629/Top-scientists-start-to-examine-fiddle-global-warming-figures.html。这种质疑与我们本节的工作无关，必须接受并使用这份数据。我抓下的数据时间跨度为1919年3月~2013年，用来匹配二氧化碳的数据。

全球二氧化碳排放量的估计可以在美国能源部二氧化碳信息分析中心的网站（http://cdiac.ornl.gov/）上找到。

我将数据放在一个.csv文件（climate.csv）中，供你下载并保存到工作目录。文件地址为https://github.com/datameister66/data/。

加载这个文件，检查数据结构：

```
> climate <- read.csv("climate.csv", stringsAsFactors = F)

> str(climate)
 'data.frame': 95 obs. of 3 variables:
 $ Year: int 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 ...
 $ CO2 : int 806 932 803 845 970 963 975 983 1062 1065 ...
 $ Temp: num -0.272 -0.241 -0.187 -0.301 -0.272 -0.292 -0.214
   -0.105 -0.208  -0.206 ...
```

将其转换为时间序列结构，指定开始年份和结束年份：

```
> climate <- ts(climate[, 2:3], frequency = 12,
   start = 1919, end = 2013)

> head(climate)
     CO2 Temp
[1,] 806 -0.272
[2,] 932 -0.241
[3,] 803 -0.187
[4,] 845 -0.301
[5,] 970 -0.272
[6,] 963 -0.292
```

加载完数据并转换为时间序列结构之后，进行分析之前的数据理解和数据准备。

数据理解与数据准备

要进行数据理解和准备，只需加载两个R包，但要确定它们已经安装在你的系统中：

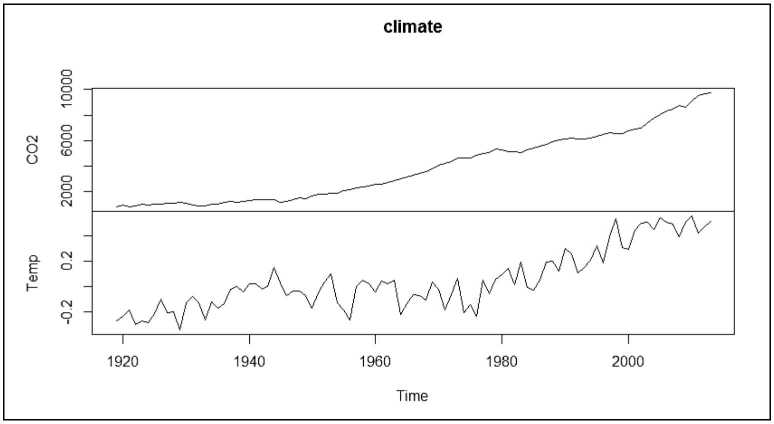
```
    > library(forecast)

> library(tseries)
```

首先绘制两个时间序列：

```
> autoplot(climate)
```

上述命令输出如下。



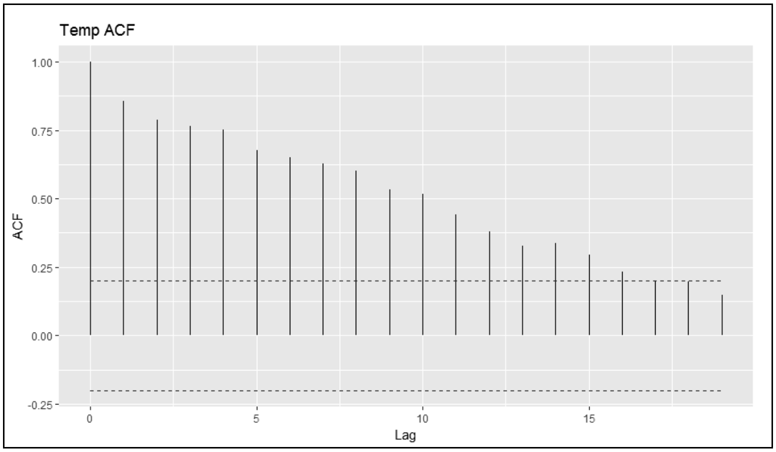
从图中可以看出，二氧化碳排放水平确实在第二次世界大战后开始增长，温度的异常变化大约发生在20世纪70年代中期，那时开始急剧升高。没有明显的异常值，不同时间的方差看上去也是个固定值。通过标准的分析过程可以知道，两个序列是高度相关的，如下所示：

```
> cor(climate)
           CO2      Temp
CO2  1.0000000 0.8404215
Temp 0.8404215 1.0000000
```

如前所述，还不到庆祝的时候，因为这根本说明不了什么问题。绘制两个序列的ACF图和PACF图，再看看序列结构：

```
> autoplot(acf(climate[, 2], plot = F), main="Temp ACF")
```

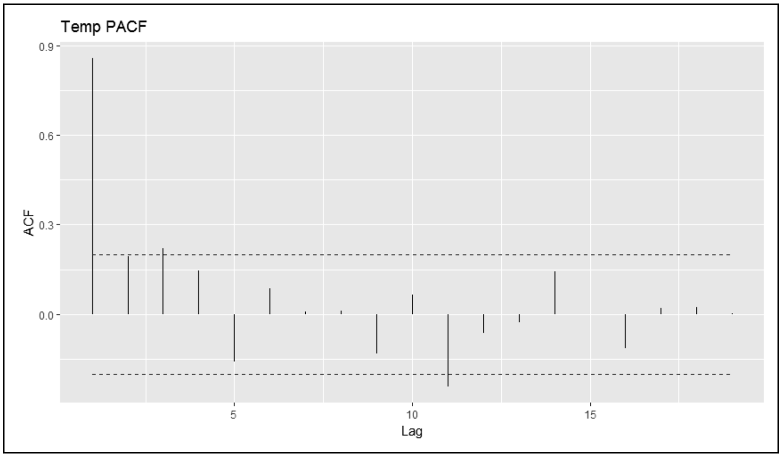
上述代码片段输出如下。



下面的代码可以绘制温度的PACF图：

```
> autoplot(pacf(climate[, 2], plot = F), main = "Temp PACF")
```

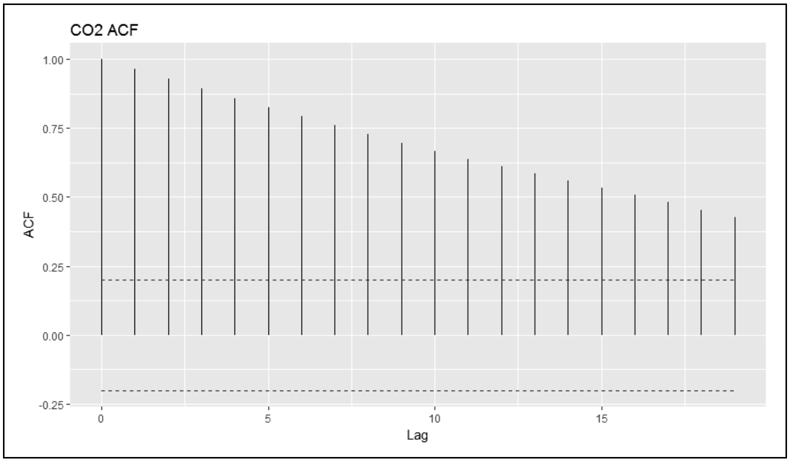
上述代码片段输出如下。



这行代码绘制二氧化碳排放量的ACF图：

```
> autoplot(acf(climate[, 1], plot = F), main = "CO2 ACF")
```

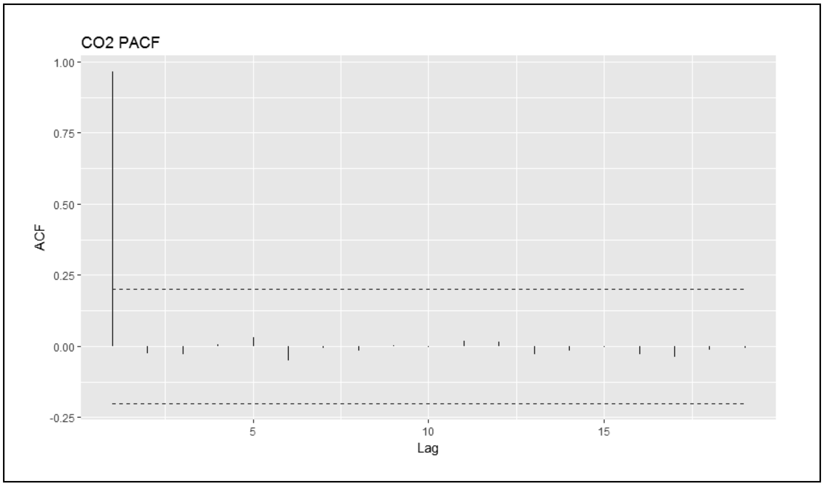
上述代码片段输出如下。



这行代码绘制二氧化碳排放量的PACF图：

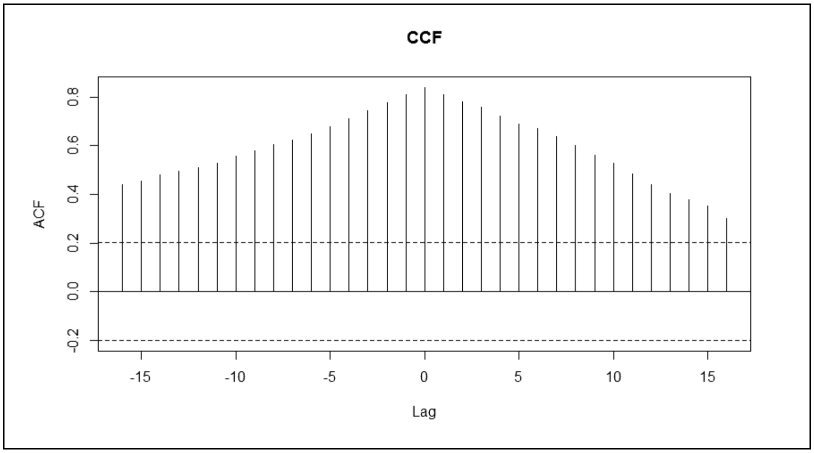
```
> autoplot(pacf(climate[, 1], plot = F), main = "CO2 PACF")
```

上述代码片段输出如下页图。



ACF模式是逐渐衰减的，PACF模式则是快速衰减的。可以假设这两个序列都是自回归的，尽管温度中看上去存在明显的MA项。下一步检查**交叉相关函数**。请注意，在函数中，要把*x*放在*y*的前面：

```
> ccf(climate[, 1], climate[, 2], main = "CCF")
```



CCF图展示了温度和二氧化碳延迟序列之间的相关性。如果*x*变量的负延迟序列具有强相关性，则*x*领先于*y*；如果*x*变量的正延迟序列具有强相关性，则*x*滞后于*y*。此处可以看到，二氧化碳既是个领先变量，也是个滞后变量。对于这个分析，我们乐于见到前者，对于后者就比较头疼。我们会在VAR（向量自回归）和格兰杰因果关系分析过程中确定这个问题是否严重。

此外还要检验数据是否平稳。可以通过tseries包中提供的**扩展迪基–福勒检验**实现这个操作，使用`adf.test()`函数，如下所示：

```
> adf.test(climate[, 1])

       Augmented Dickey-Fuller Test

data: climate[, 1]
Dickey-Fuller = -1.1519, Lag order = 4, p-value =
0.9101
alternative hypothesis: stationary

> adf.test(climate[, 2])

       Augmented Dickey-Fuller Test

data: climate[, 2]
Dickey-Fuller = -1.8106, Lag order = 4, p-value =
0.6546
alternative hypothesis: stationary
```

可以看到，对于这两个序列，p值都是不显著的，所以不能拒绝原假设（数据是不稳定的）。

进行数据探索之后，下面开始建模阶段，首先对温度异常数据使用单变量分析技术。

12.3　模型构建与模型评价

模型构建与评价这一步骤主要有三项任务：第一，建立一个仅应用于温度数据的单变量预测模型；第二，基于温度数据本身和二氧化碳排放量数据建立一个温度数据的回归模型；第三，使用这个模型的输出揭示二氧化碳排放量和地表温度异常之间是否存在格兰杰因果关系。

12.3.1　单变量时间序列预测

这项任务的目标是为地表温度建立一个单变量预测模型，重点在于选择Holt线性趋势模型还是ARIMA模型。就像在其他学习技术中做的那样，我们要训练模型，并确定模型在测试集上的预测准确度，测试集与训练集使用不同的时间序列。下面的代码创建了温度数据子集以及训练集和测试集，时间从二战后开始：

```
> temp <- climate[, 2]

> temp <- climate[, 2]

> train <- window(temp, start = 1946, end = 2003)

> test <- window(temp, start = 2004)
```

为了建立平滑模型，可以使用forecast包中的`holt()`函数。我们要建立2个模型，一个有阻尼趋势，另一个则没有。在这个函数中，需要指定时间序列、预测阶段数量h，以及选择初始状态值的方法（`"optimal"`或者`"simple"`），还有是否需要阻尼趋势。选择初始状态值时，`"optimal"`

表示算法会找出最优的初始值以及平滑参数，"simple"表示使用开始的几个观测来计算初始值。在forecast包中可以使用ets()函数，它能够找出所有最优参数。但在这个案例中，我们依然使用holt()函数，因为这样可以进行比较。现在，建立没有阻尼趋势的holt模型，如下所示：

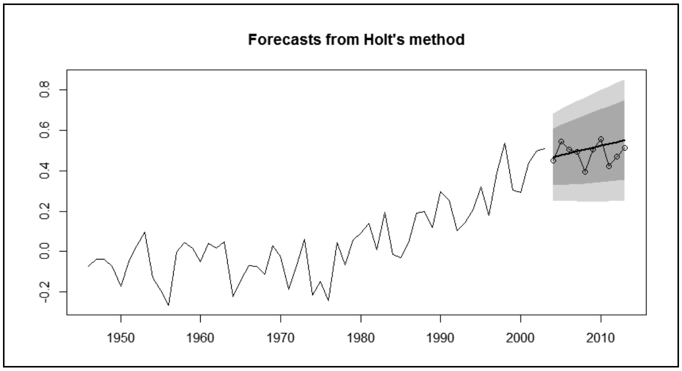
```
> fit.holt <- holt(train, h = 10, initial = "optimal")
```

绘制预测值，看看模型在样本外数据上的预测效果：

```
> plot(forecast(fit.holt))

> lines(test, type = "o")
```

上述代码输出如下。



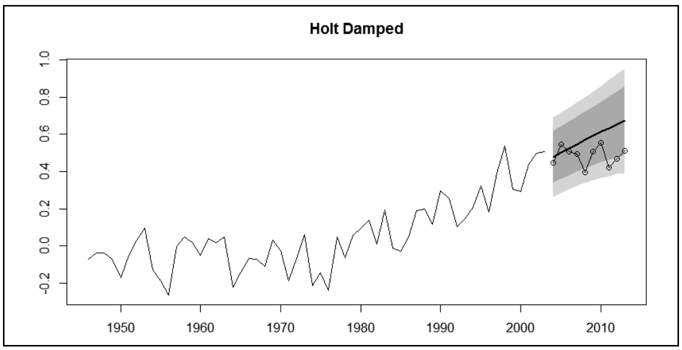
看上去，这张图中的预测值表现出轻微的线性上升趋势。下面加上阻尼趋势，如下所示：

```
> fit.holtd <- holt(train, h = 10, initial = "optimal", damped =
  TRUE)

> plot(forecast(fit.holtd),main = "Holt Damped")

> lines(test, type = "o")
```

上述代码输出如下。



在单变量分析的最后，建立一个ARIMA模型，使用forecast包中的`auto.arima()`函数。这个函数中有很多选项可以设置，然而你可以仅指定时间序列数据，它会自己找到最优的ARIMA拟合：

```
> fit.arima <- auto.arima(train)
> summary(fit.arima)
Series: train
ARIMA(0,1,1) with drift

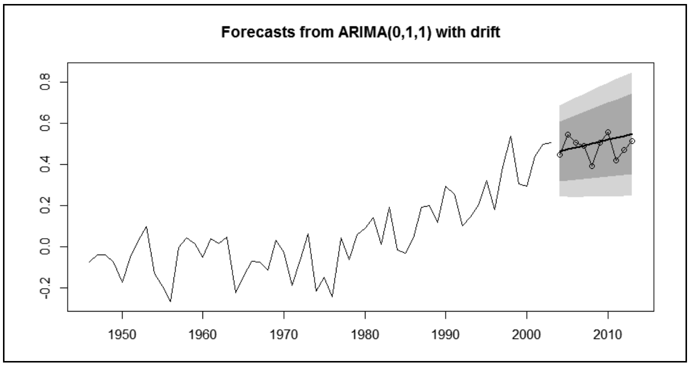
Coefficients:
         ma1   drift
     -0.6949  0.0094
s.e.  0.1041  0.0047
```

从简化过的输出可知，函数选择的模型是MA = 1、I = 1，也就是带有漂移项（等价于截距）的ARIMA(0, 1, 1)。和前面一样，可以通过图形检查它在测试数据上的表现：

```
> plot(forecast(fit.arima, h = 10))

> lines(test, type="o")
```

上述代码输出如下。



这个图与不带阻尼趋势的holt方法非常相似。通过下面的代码，可以为每种模型打分，找出具有最低误差，即**平均绝对百分误差**的模型：

```
> mapeHOLT <- sum(abs((test - fit.holt$mean)/test))/10

> mapeHOLT
[1] 0.105813

> mapeHOLTD <- sum(abs((test - fit.holtd$mean)/test))/10

> mapeHOLTD
[1] 0.2220256

> mapeARIMA <- sum(abs((test - forecast(fit.arima, h =
```

```
 10)$mean)/test))/10

> mapeARIMA
[1] 0.1034813
```

与holt方法相比，ARIMA(0, 1, 1)的预测误差要稍小一些。很显然，带有阻尼趋势的模型表现最差。

通过统计检验和可视化证据，单变量预测模型的最好选择似乎是ARIMA模型。

这样就完成了对地表温度异常单变量预测模型的构建，然后进行下一项任务，看看二氧化碳排放水平是否会引起这些异常。

12.3.2 检查因果关系

我认为本节最能体现序列分析技术的价值，我们会将因果关系与相关性区别开来，当然，此处的因果关系是统计意义上的。我们不是第一个将格兰杰因果关系应用于碳排放问题的研究团队。Triacca（2005）发现，没有明显的证据能够证明大气中的二氧化碳量与地表温度异常之间存在格兰杰因果关系。而另一方面，Kodra（2010）的结论则认为二者之间存在因果关系，但他同时发表了一个附加说明，提示他的数据是不平稳的，甚至在二阶差分之后也是如此。我们的努力不是为了平息争论，而是希望鼓励你将方法理论应用于实际工作。本节内容完全可以为格兰杰因果关系模型提供一次有效的实战练习。

我们的计划是，首先演示残差受到自相关（也称为序列相关）影响而导致的虚假线性回归。然后，研究两种实现格兰杰因果关系模型的不同方法。第一种是传统方法，其中两个序列都是平稳的。此后看看Toda和Yamamoto（1995）提出的方法，这种方法可以应用于原始数据（有时也称为“水平”）。

1. 线性回归

我们从虚假线性回归开始。在现实世界中，这种情况我已经司空见惯。下面简单建立一个线性模型，并检查结果：

```
> fit.lm <- lm(Temp ~ CO2, data = climate)

> summary(fit.lm)

Call:
lm(formula = Temp ~ CO2, data = climate)

Residuals:
     Min       1Q   Median       3Q      Max
-0.36411 -0.08986 0.00011 0.09475 0.28763

Coefficients:
              Estimate Std. Error  t value    Pr(>|t|)
```

```
(Intercept) -2.430e-01  2.357e-02   -10.31  <2e-16 ***
CO2          7.548e-05  5.047e-06    14.96  <2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

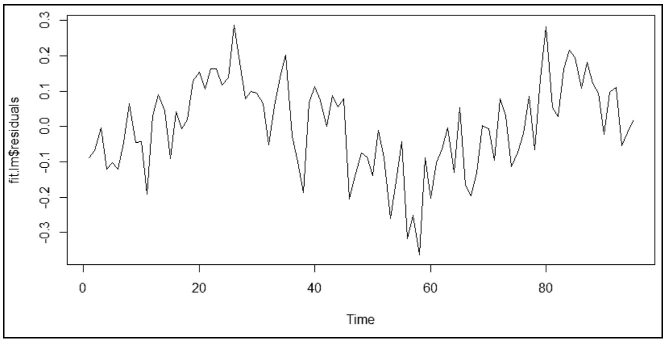
Residual standard error: 0.1299 on 93 degrees of freedom
Multiple R-squared: 0.7063, Adjusted R-squared: 0.7032
F-statistic: 223.7 on 1 and 93 DF, p-value: < 2.2e-16
```

请注意，所有统计量都是显著的，修正R方的值是0.7。好吧，二氧化碳水平与温度变化高度相关，但是根据Granger和Newbold（1974）的论述，这一切都没什么意义。再说一次，我见过很多具有高学位的人在会议上使用这样的结果自吹自擂，每次我都忍不住做“坏人”，对这种结果提出异议。

可以绘制序列相关性，首先是残差的时间序列图，能够看出残差具有明显的模式：

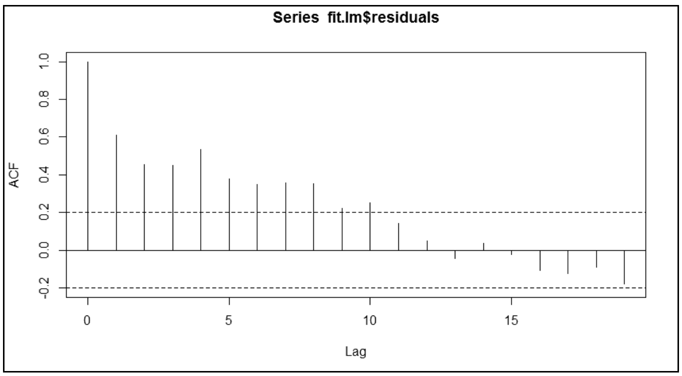
```
> plot.ts(fit.lm$residuals)
```

上述代码输出如下。



然后创建一幅ACF图，可以看出，直到10阶延迟序列，都具有显著的自相关：

```
> acf(fit.lm$residuals)
```



可以对自相关性使用Durbin-Watson检验。这个检验的原假设是“不存在自相关”：

```
> dwtest(fit.lm)

Durbin-Watson test

data: fit.lm
DW = 0.77425, p-value = 4.468e-12
alternative hypothesis: true autocorrelation is greater than 0
```

从对图形的检查可以知道，这个结果没有什么可奇怪的，我们完全可以拒绝“没有自相关性”的原假设。处理自相关的简单方式是，在相关的时间序列中加入延迟变量，使所有数据具有稳定性。下面进行这样的处理，使用向量自回归找出适当的延迟结构，加入我们的因果关系模型。

2. 向量自回归

通过前面的章节我们知道，温度数据是平稳的，二氧化碳数据则需要一阶差分。另一种可以表示这种情况的简单方法是，使用forecast包提供的`ndiffs()`函数。从它的输出中可以找出使数据平稳所需的最小差分次数。在这个函数中，你可以选择3种可用的检验方法之一，这3种检验方法分别是：**Kwiatkowski, Philips, Schmidt & Shin**（**KPSS**）、**Augmented Dickey Fuller**（**ADF**）和**Philips-Peron**（**PP**）。在下面的代码中，我将使用ADF方法，它的原假设认为数据是不平稳的：

```
> ndiffs(climate[, 1], test = "adf")
[1] 1

> ndiffs(climate[, 2], test = "adf")
[1] 1
```

在两个序列中，一阶差分即可使数据稳定。首先建立差分，然后完成传统方法，其中两个序列都是稳定的。要进行这个练习，依然要先加载软件包：

```
> library(vars)

> library(aod)

> climateDiff <- diff(climate)

> climateDiff <- window(climateDiff, start = 1946)

> head(climateDiff)
      CO2     Temp
[1,]   78   -0.099
[2,]  154    0.034
[3,]   77    0.001
[4,]  -50   -0.035
[5,]  211   -0.100
[6,]  137    0.121
```

现在的重点是，使用向量自回归在信息准则的基础上确定最优的延迟结构，可以使用vars包

中的`VARselect()`函数完成。只需在函数中指定数据，并用`lag.max = x`指定模型中延迟的最大数量即可。我们设定最大延迟数量为12：

```
> lag.select <- VARselect(climateDiff, lag.max = 12)

> lag.select$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
     5      1      1      5
```

可以使用lag$selection调用信息准则，有4种不同的信息准则：**AIC**、**Hannan-Quinn Criterion**（**HQ**）、**Schwarz-Bayes Criterion**（**SC**）和**FPE**。请注意，第2章已经介绍过AIC和SC，所以此处不再具体介绍它们的公式和区别。如果你想看看每种延迟的实际结果，可以使用lag$criteria。能够看到，AIC和FPE都选择了5阶延迟作为VAR模型的最优结构，HQ和SC则选择了一阶延迟。看上去应该使用5年的延迟，我们使用`var()`函数建立这个5阶延迟模型。一阶延迟模型就交给你了：

```
> fit1 <- VAR(climateDiff, p = 5)
```

上述代码的摘要结果非常长，因为它建立了两个单独的模型，这个结果可能会占满两页纸。我在这里提供的是一个简化的输出，表示温度的预测结果：

```
> summary(fit1)
Residual standard error: 0.1006 on 52 degrees of freedom
Multiple R-Squared: 0.4509, Adjusted R-squared: 0.3453
F-statistic: 4.27 on 10 and 52 DF, p-value: 0.0002326
```

模型是显著的，修正R方为0.35。

和前一节一样，需要检查序列相关性。VAR包提供了`serial.test()`函数进行多变量自相关检验，它可以进行几种不同的检验，我们重点进行Portmanteau检验。还要注意，DW检验只适用于单变量序列。Portmanteau检验的原假设认为自相关为0，备择假设认为自相关不为0：

```
> serial.test(fit1, type = "PT.asymptotic")

    Portmanteau Test (asymptotic)

data: Residuals of VAR object fit1
Chi-squared = 35.912, df = 44, p-value = 0.8021
```

0.3481的p值让我们没有证据拒绝原假设，可以认为残差中不存在自相关。一阶延迟模型的检验结果如何？

要在R中进行格兰杰因果关系检验，可以使用lmtest包提供的`Grangertest()`函数，也可以使用vars包提供的`causality()`函数。我将演示如何使用`causality()`函数。非常简单，因为你只需要建立两个对象，一个表示x引发y，一个表示y引发x，然后使用前面生成的fit1对象：

```
> x2y <- causality(fit1, cause = "CO2")

> y2x <- causality(fit1, cause = "Temp")
```

现在，进行格兰杰检验将非常简单：

```
> x2y$Granger

     Granger causality H0: CO2_diff do not Granger-cause
     climate2.temp

data: VAR object fit1
F-Test = 2.2069, df1 = 5, df2 = 104, p-value = 0.05908

> y2x$Granger

     Granger causality H0: climate2.temp do not Granger-cause
     CO2_diff

data: VAR object fit1
F-Test = 0.66783, df1 = 5, df2 = 104, p-value = 0.6487
```

“二氧化碳变化是引发温度变化的格兰杰原因”的p值为0.05908，另一个方向则是不显著的。这能说明什么问题呢？首先，可以认为y不是引发x的原因。至于x引发y，我们不能在0.05的显著性水平上拒绝原假设。所以，可以得出结论：x不是y的格兰杰原因。但是，这就是最终结论吗？别忘了，p值的意义就是当原假设为真时出现样本情况的概率。还有，我们的检验从来没有设计为那种非此即彼的情况。如果这是一个受控实验，比如美国食品与药物管理局进行的三阶段临床实验，那么我们可以毫不犹豫地宣布没有足够的证据拒绝原假设。因为我们的研究是基于观测性数据的，所以我相信，可以认为“二氧化碳排放量非常可能是地表温度异常的格兰杰原因”。尽管如此，这个结论还是给批评者留下了很大的空间，比如前面提到的关于数据质量的争议。还在困扰我的一件事是，应该从哪一年开始分析。我选择1945年是因为看上去应该如此，用SAS的术语来说，这是调用了一个“眼球过程”。选择从哪一年开始对我们的分析具有巨大的影响：这会改变延迟结构，还会导致不显著的p值。

但是，我们还需要使用另外一种格兰杰因果关系技术来为初始的二氧化碳排放数据建模。找到正确延迟数量的过程和前面基本一样，区别在于不需要使数据平稳：

```
> climateLevels <- window(climate, start = 1946)

> level.select <- VARselect(climateLevels, lag.max = 12)

> level.select$selection
AIC(n) HQ(n) SC(n) FPE(n)
    10     1     1      6
```


尝试一下6阶延迟结构，看看能否得到显著的结果，记得要加上一个额外的延迟来表示差分序列。对这种技术的讨论以及需要使用这种技术的原因可以参考http://davegiles.blogspot.de/2011/04/testing-for-granger-causality.html。

```
  fit2 <- VAR(climateLevels, p = 7)
> serial.test(fit2, type = "PT.asymptotic")
```

```
        Portmanteau Test (asymptotic)

data: Residuals of VAR object fit2
Chi-squared = 35.161, df = 36, p-value = 0.5083
```

现在，为了确定x是引起y的格兰杰原因，需要进行Wald检验。在预测y的公式中，有且只有x的系数为0。请记住，不要在测试中包括表示差分的其他系数。

R语言的Wald检验在aod包中，我们已经加载过。需要指定整个模型的系数、方差–协方差矩阵和因果关系变量的系数。

在VAR对象中，需要检验的温度系数是2~12的偶数，而二氧化碳系数是1~11的奇数。函数不使用`c(2, 4, 6, ...)`这种形式，而是使用基础R包中的`seq()`函数创建一个序列对象。

首先，看看二氧化碳排放量是否是温度变化的格兰杰原因：

```
> CO2terms <- seq(1, 11, 2)

> Tempterms <- seq(2, 12, 2)
```

现在进行Wald检验，代码如下所示：

```
> wald.test(b = coef(fit2$varresult$Temp),
Sigma = vcov(fit2$varresult$Temp),
Terms = c(CO2terms))
   Wald test:
   ----------

Chi-squared test:
X2 = 11.5, df = 6, P(> X2) = 0.074
```

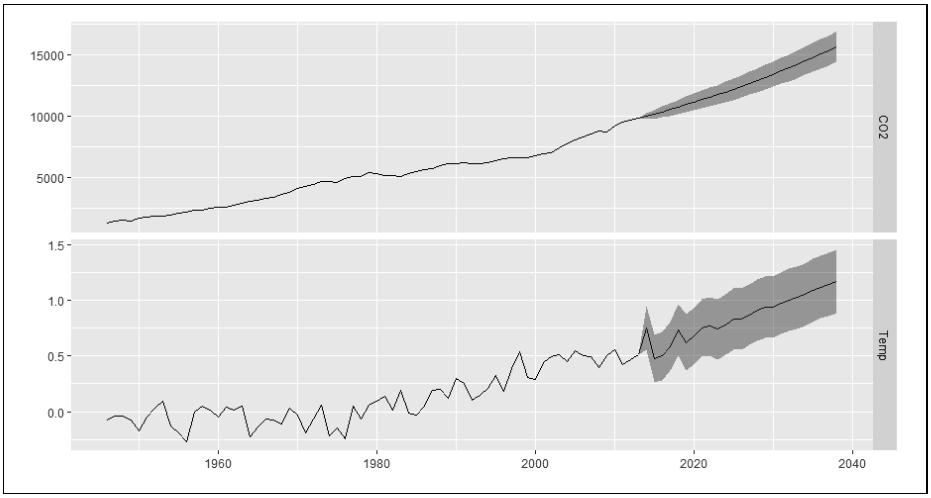
结果如何？非常接近边际p值0.05。再检验因果关系的另一个方向，代码如下：

```
> wald.test(b = coef(fit2$varresult$CO2),
Sigma = vcov(fit2$varresult$CO2),
Terms = c(Tempterms))
   Wald test:
   ----------

Chi-squared test:
X2 = 3.9, df = 6, P(> X2) = 0.69
```

最后一件事是如何使用向量自回归进行预测。`predict()`函数已经准备好，对其应用`autoplot()`函数，绘制阶段为25年的预测，看看会是什么情况：

```
> autoplot(predict(fit2, n.ahead = 25, ci = 0.95))
```



前途一片灰暗，按照流行剧集《权力的游戏》中的话来说就是——“凛冬将至”。我其实无所谓，因为我的长期投资和储蓄计划中已经包括罐头和各种军用物资了。我还能怎么做呢？骑马出去上班吗？如果阿尔·戈尔能这么做，我也可以。在此期间，我可能要擦着防晒油上班了。

闲话少说，我希望本章的讨论可以促使你思考，如何应用技术去解决实际问题，甚至可以将气候变化数据研究得更深入一些。揭示因果关系需要非常艰苦的努力，格兰杰因果关系在这方面是一个非常好的工具。

12.4 小结

本章目的是讨论时间因素在机器学习与分析中多么重要，同时指出分析时间序列时经常陷入的误区，并介绍了走出误区的技术和方法。我们介绍了单变量和双变量时间序列分析，并使用这两种方法研究了全球温度异常和人类二氧化碳排放量之间的关系。此外还介绍了格兰杰因果关系模型，并用来确定是否可以在统计意义上认为二氧化碳排放量会引起地表温度异常。我们发现，“二氧化碳排放量是温度异常的格兰杰因果关系”的p值大于0.05，但小于0.1。这说明在机器学习领域内的因果性研究中，格兰杰因果关系模型这个工具相当有效。下一章研究如何将机器学习方法应用于文本数据。

此外，请记住我们只学习了时间序列分析的皮毛。我希望你能研究更多技术，比如突变点检测、时间序列分解、非线性预测等。尽管这些不是机器学习中的常用技术，但我相信它们可以使你的能力得到极大提高。

第 13 章 文本挖掘

“我觉得，不知道答案的生活要比得到一个错误答案有趣得多。”

——理查德·费曼

现在这个世界，文本数据已经泛滥成灾。如果你使用谷歌、必应或雅虎查询有多少数据是非结构化的，也就是文本形式的，估计这个比例会有80%～90%。具体的比例数据并不重要，重要的是大部分数据都是文本形式的。这说明任何一个想在数据中淘到宝藏的人，都必须具备处理和分析文本数据的能力。

我刚开始从事市场分析时，经常一页页翻看小组访谈和采访记录，期望能够洞察一些独家信息（就是那种“啊哈！”时刻），然后和同事们神侃，看看他们是否也能有同样的发现。然而，项目组中总有那么一个家伙，在没人注意的时候，不知道从哪里跑出来，在听了两次访谈（时间过去三四十分钟）之后，才恍然大悟，明白我们究竟在干什么。现在的情况比那时好多了，当前的技术可以使分析师从数据中快读提取有意义的定量结果，从而支持对问题的定性理解，甚至可以帮助那些中途加入的人。

在过去的几年中，我使用文本挖掘技术做了很多工作，包括对医生与患者之间的交流记录进行挖掘，理解美国食品和药物管理局对于处方药广告的担忧，捕获患者对于罕见癌症的关注，等等。使用R和本章中的方法，你可以从文本数据中提取非常有用的信息。

13.1 文本挖掘框架与方法

有很多种方法可以进行文本挖掘。本节的目的是提供文本挖掘的基础框架，这个框架并不能包括所有可用的算法，但能够覆盖你遇到的大多数项目中的最重要的内容。此外，因为文本挖掘的建模可能会非常复杂，所以我会尽量以简明扼要的方式讨论建模方法。“如何收集和整理文本数据”这个主题需要用好几章的篇幅进行介绍，所以我们假设使用的数据来自Twitter、客户呼叫中心、网络信息收集或其他任何可以包含文本文件的对象。

首要任务是将文本文件放入一个结构化的文件中，这个文件称为**语料库**。语料库中的文档可

以只有一个，也可以有几十、几百甚至几千个。R可以处理的原始文本文件包括RSS源文件、PDF文件和微软Word文档。建立语料库后，可以通过文本转换进行数据准备。

下面列出了文本文件转换时最常用、最有效的操作：

- 将大写字母转换为小写字母；
- 剔除数字；
- 剔除标点符号；
- 剔除停用词；
- 剔除多余的空白字符；
- 词干提取；
- 词语替换。

进行语料库转换时，你不仅要建立一个更紧凑的数据集，还要简化文档结构，以便更容易发现词语之间的关系，从而加深对数据的理解。但是请记住，在任何情况下，这些转换操作都不是必需的，应当根据实际情况判断需要哪些转换。有时还要反复进行判断，以找到最有意义的转换。

将词语转换成小写可以防止对词语的错误计数。假设你对hockey这个词的计数为3，但当它是句子的第一个词时，你又会对Hockey这个词计数为1。R不会给出`hockey = 4`的计数结果，而会得出`hockey = 3`以及`Hockey = 1`。

剔除标点符号也是出于同样的目的。但在后面的实际案例中会看到，如果你想把文档分成句子，标点符号还是非常重要的。

剔除停用词是指，将那些没有价值的常见词去掉，实际上，它们不利于分析是因为出现的频率过高，会掩盖真正重要的词语。常见的停用词有are、and、is、the、not和to。剔除空白字符可以使语料库更加紧凑，此类字符通常有制表符、段落标记、双倍行距标记等。

词干提取会有一点棘手，因为它删除了词的后缀，生成了称为**词根**的基本词语，这可能会产生一些混淆。我个人对于词干提取不太感冒，与我一起工作的分析师也同意这个观点。尽管如此，可以使用R包中提供的`tm`函数进行词干提取，函数使用的是SnowballC包中的**波特词干提取算法**。假如你的语料库中有family和families两个词，R会分别为二者计数。运行词干提取算法后，这两个词都会变为famili。这样会防止错误计数，但有些时候的结果解释看上去会怪怪的。比如做展示时，词云中如果出现这种词，就显得不那么动人了。某些情况下应该做两次分析，一次使用提取了词干的词，另一次使用未提取词干的词，看看哪种更有意义。

词语替换这种转换可做可不做。词语替换的目的是将具有相似意义的词组合在一起，例如management和leadership。你可以使用词语替换来代替词干提取。我研究过进行词干提取和不进行词干提取的结果，个人认为，如果不进行词干提取而是替换掉一组词的话，可以得到更有意义的结果。

语料库转换完成后，下一步就是建立**文档–词矩阵**或**词–文档矩阵**。这两个矩阵中保存的都是某个词在某个文档中出现的次数，前者中，行表示文档，列表示词；而在后者中，行表示词，列表示文档。这两种矩阵都可以用于文本挖掘。

使用上面的矩阵即可开始文本分析，你可以研究词频，也可以生成可视化结果，比如词云。生成特定词语的相关性列表可以找出词语关联。文档–词矩阵还是用来建立主题模型的一种必要的数据结构。

13.2　主题模型

主题模型是按照文档主题为文档分组的一种强有力的方法。主题模型允许对文档中的名词出现频率进行概率建模。拟合后的模型可以用来估计文档之间或者一组特定关键词之间的相似度，模型使用额外一层潜在变量进行估计，这些潜在变量就称为主题（Grun与Hornik，2011）。从本质上讲，一篇文档是根据其内部词语的分布来分配给一个主题的，这个主题下的其他文档也具有几乎相同频率的词语。

我们重点讨论的算法是**隐含狄利克雷分布**，它使用的是吉布斯抽样方法，这可能是最常用的一种抽样方法。建立主题模型时，主题的数量必须在运行算法之前就确定下来（*k*维）。如果没有先验理由可以确定主题数量，那么可以建立若干模型，然后使用专业知识和判断力做出最后的选择。带有吉布斯抽样的LDA用数学解释起来非常复杂，但既然我们的目的是介绍主题模型，所以你至少应该可以通俗地描述算法如何学会将一篇文档分配给一个主题。如果你想弄清楚数学解释，那么就抽出几个小时去尝试一下吧。https://www.cs.princeton.edu/~blei/papers/Blei2012.pdf提供了很详细的背景资料。

LDA是一个生成式过程，它按照下面的步骤进行迭代，直到达成一个平稳状态。

(1) 如果有1 ~ *j*个文档，1 ~ *k*个主题，那么对每个文档（*j*）使用一个多项式分布（狄利克雷分布）将其随机分配给各个主题（*k*）。例如，文档A分配给主题1的概率是25%，分配给主题2的概率是25%，分配给主题3的概率是50%。

(2) 如果有1 ~ *i*个词，那么每个词都以某个概率属于某个主题（*k*）。例如，词mean以0.25的概率属于主题statistics。

(3) 对于文档（*j*）和主题（*k*）中的每一个词（*i*），计算该文档中的词分配给该主题的比例，记为主题（*k*）在文档（*j*）中的概率P(*k*|*j*)。再从所有包含该词的文档中计算词（*i*）在主题（*k*）中的比例，记为词（*i*）属于主题（*k*）的概率P(*i*|*k*)。

(4) 重新抽样，即基于*t*包含*w*的概率为*w*分配一个新的*t*，此处的基准概率为P(*k*|*j*) × P(*i*|*k*)。

(5) 重复这一过程，经过多次迭代，算法会最终收敛。于是，基于每个词被分配给文档中主题的比例，这篇文档就被分配给一个主题。

本章中的LDA假设词语和文档的顺序并不重要。现在已经有研究放松了这种假设，以建立语

言生成模型和随着时间变化的序列模型（称为**动态主题建模**）。

其他定量分析

本节论的内容是基于句子对文本进行语义分析，以及从语言学的角度对词语进行标记，比如名词、动词、代词、形容词、副词、介词、单数、复数等。一般来讲，只研究词频和文本隐含主题就足够了。但是，有些时候你会发现需要对文本风格理解得更深入一些，以便对演讲者和写作者进行比较。

有很多方法可以完成这一任务，但我们只集中讨论下面5种：

- 极性分析（情感分析）
- 自动易读性指数（复杂度）
- 正式度
- 多样性
- 分散度

极性分析通常称为情感分析，它可以告诉你文本的情感有多么积极或者多么消极。R使用qdap包进行极性分析，它为每个句子分配一个评分。你可以对不同作者、文本或主题进行分组，以分析极性均值或标准差。有很多极性词典可供使用，qdap包中默认使用的词典是由Hu与Liu在2004年建立的。你可以根据自己的需要替换或改变这个词典。

算法首先对词语进行标记，每个词都根据词典标为“积极”“消极”或“中性”的情感标记。标记后的词与它前4个词和后2个词一起，被聚集成词簇。词簇用**效价转换器**（valence shifter，中性器neutral、否定器negator、放大器amplifier、负放大器de-amplifier）进行标记。根据词和词簇的数量与位置，为其赋予一系列权重，然后将权重相加，再除以句子中词的数量的平方根。

自动易读性指数是衡量文本复杂度和读者理解能力的一个指标。可以用一个专门的公式计算数：4.71(字符数/词数) + 0.5(词数/句子数) – 21.43。

这个指数会生成一个数值，通过这个数值可以估计出能完全理解文本的学生年级。如果这个数值是9，那么一个年龄为13~15岁的学生应该能够理解文本意义。

正式度可以表示文本和读者之间或者演讲者与听众之间的相关程度。我认为，它是表示文本作者与读者之间适合程度的一种方式，或者是对作者与读者交流环境的一种设定。如果你想体验正式的文本，那么去参加医学会议或阅读法律文件吧。非正式文本天然地要受到语境的影响。

正式度可以用**F值**度量，这个度量值的计算方法如下所示。

- 正式词（f）：名词、形容词、介词、条款
- 上下文关系词（c）：代词、动词、副词、感叹词

- N = (f + c + 连接词)的总数
- 正式度指数 = 50((f的总数 – c的总数/N) + 1)

这些都无关紧要，但当我在伊拉克时，需要为一位将军（名字还需保密）总结并撰写形势分析报告。这位将军坚决禁止在报告中使用副词，否则他就会勃然大怒。他的理由是，“非常”或“大多数”这种词是不能量化的，因为不同的人对此有不同理解。5年过去了，我却还在使用那些不必要的副词来粉饰自己的商业邮件和PPT。这就是正式度的体现！

在文本挖掘中，**多样性**表示文本中使用的不同词数和全部词数的比值。它还可以表示文本作者的词汇范围或词汇的丰富程度。qdap包提供了5种（你没看错，是5种）不同的多样性度量：Simpson、Shannon、Collision、Bergen Parker、和Brillouin。我不会详细介绍这5种度量方式，只是想告诉你，这种算法不只用于通信和信息科学检索，还可以用于分析自然界中的生物多样性。

最后看看**分散度**，或称**词汇分散度**。它是一种可以帮助你理解词在整篇文本中的分布的有用工具，也是探索文本并识别模式的极好方法。进行分散度分析时，首先找出特定的词或词组，然后在统计图中展现该词或词组出现于文本的时间。在后面会看到，qdap包提供了一个内建的制图函数，以帮助分析文本分散度。

我们介绍了一个文本挖掘的框架，可以用来做文本准备、词汇计数以及主题模型建模，之后对其他一些词汇指标进行了深入探讨。现在，将上述理论应用于现实中的文本挖掘问题。

13.3　业务理解

在这个案例中，我们研究奥巴马前总统的国会演讲。我没有什么预定目标，仅仅好奇能否从中发现一些特别的东西，还想知道随着时间的推移，他传达的信息有什么变化。这个案例有可能成为分析政客演讲的一个蓝本，可以用于准备辩论中的反对意见，也可以生成自己的演讲稿。如果不可行，那也没什么大不了的。

我们有两个主要的分析目标，首先使用7篇国会演讲建立一个主题模型，然后对比2010年的第一篇演讲和2016年1月的最后一篇演讲。使用基于句子的文本度量方式，比如情感和分散度。

数据理解与数据准备

我们要使用的基础软件包是tm，这是专门的文本挖掘软件包。还需要SnowballC包进行词干提取，使用RColorBrewer包为词云上色，当然还有wordcloud包。在加载之前，请确认你已经安装了它们：

```
    > library(tm)

> library(wordcloud)

> library(RColorBrewer)
```

可以从https://github.com/datameister66/data下载数据文件。请一定将这些文本文件放在一个单独的目录中，因为我们会使用这个目录作为语料库进行分析。

需要下载7个.txt文件，比如sou2012.txt，将这些文件放在你的R语言工作目录中。可以用以下命令找出你当前的工作目录，或者设置工作目录：

```
> getwd()

> setwd(".../data")
```

现在可以建立语料库了。首先建立一个包含演讲稿文件路径的对象，然后看看目录中有多少个文件，以及每个文件的名字：

```
> name <- file.path(".../text")

> length(dir(name))
[1] 7

> dir(name)
[1] "sou2010.txt" "sou2011.txt" "sou2012.txt" "sou2013.txt"
[5] "sou2014.txt" "sou2015.txt" "sou2016.txt"
```

将语料库命名为docs，并使用`Corpus()`函数建立语料库。函数包装了`DirSource()`函数，它也是tm包的一部分：

```
> docs <- Corpus(DirSource(name))

> docs
<<VCorpus>>
Metadata: corpus specific: 0, document level (indexed): 0
Content: documents: 7
```

请注意，语料库中没有corpus和document level元数据。tm包中有些函数可以生成作者名和时间戳这样的信息，也能生成document level和corpus。这个案例中不需要这些信息。

现在，使用tm包中的`tm_map()`函数进行文本转换。前面讨论过，需要执行的转换操作有：字母转换为小写、剔除数字、剔除标点符号、剔除停用词、剔除空白字符，然后进行词干提取。

```
> docs <- tm_map(docs, tolower)

> docs <- tm_map(docs, removeNumbers)

> docs <- tm_map(docs, removePunctuation)

> docs <- tm_map(docs, removeWords, stopwords("english"))

> docs <- tm_map(docs, stripWhitespace)
```

此时，应该删除那些不必要的词。例如，在演讲过程中，当国会议员对于某个部分鼓掌欢呼的时候，文本中就会出现`(Applause)`，这是必须删除的：

```
> docs <- tm_map(docs, removeWords, c("applause", "can", "cant",
  "will",
"that", "weve", "dont", "wont", "youll", "youre"))
```

结束文本转换和冗余词删除之后，请确认文档还是纯文本形式，然后将其放入文档–词矩阵并查看维度：

```
    > docs = tm_map(docs, PlainTextDocument)

> dtm = DocumentTermMatrix(docs)

> dim(dtm)
[1]    7 4738
```

这7篇演讲稿包含4738个词。你可以通过`removeSparseTerms()`函数删除稀疏项，但这一步不是必需的。你需要指定一个0和1之间的数，这个数值越大，表示矩阵的稀疏度越高。稀疏度表示一个名词在文档中的相对频率，所以如果你的稀疏度阈值是0.75，那么就只删除那些稀疏度大于0.75的名词。在这个例子中，(1 – 0.75) * 7 = 1.75，因此，对于任何一个名词，如果包含它的文档少于2个，它就会被删除：

```
    > dtm <- removeSparseTerms(dtm, 0.75)

> dim(dtm)
[1]    7 2254
```

因为没有文档元数据，所以应该命名矩阵中的行，这样才能知道每行代表的文档：

```
> rownames(dtm) <- c("2010", "2011", "2012", "2013", "2014",
  "2015", "2016")
```

通过`inspect()`函数检查文档–词矩阵。此处看看所有7行中的前5列数据：

```
> inspect(dtm[1:7, 1:5])
      Terms
  Docs abandon ability able abroad absolutely
  2010       0       1    1      2          2
  2011       1       0    4      3          0
  2012       0       0    3      1          1
  2013       0       3    3      2          1
  2014       0       0    1      4          0
  2015       1       0    1      1          0
  2016       0       0    1      0          0
```

看上去，我们已经做好分析数据的准备了，下面从词频分析开始。我要指出的是，这个输出可以说明为什么我不喜欢进行大规模的词干提取。你可能正在考虑是否应该将ability和able合在一起。如果你对文档进行了词干提取，那么这两个词都会变成abl。这对分析有什么帮助吗？我

认为反而会丢失语境，至少在初始分析中的确如此。再说一次，我建议进行词干提取时要小心谨慎，三思而行。

13.4 模型构建与模型评价

建模过程将分为两个独立部分。在第一部分中，我们重点进行词频分析和相关性分析，然后建立一个主题模型。在第二部分中，我们使用功能强大的qdap包研究各种定量分析技术，以比较两篇演讲稿。

13.4.1 词频分析与主题模型

建立文档-词矩阵之后，开始词频分析。首先建立一个对象，计算每列总和，然后按降序重新排列。要计算每列总和，必须在代码中使用`as.matrix()`函数。默认的排序方式是升序，所以要在freq前面加一个负号，将排序方式改变为降序：

```
    > freq <- colSums(as.matrix(dtm))

> ord <- order(-freq)
```

通过下列代码检查对象的头部和尾部：

```
    > freq[head(ord)]
new  america   people  jobs  now  years
       193       174   168  163     157      148

> freq[tail(ord)]
wright written yearold youngest youngstown zero
             2       2        2          2      2    2
```

出现最频繁的词是new，你可能已经想到了，总统先生也非常频繁地提起america。还应该注意到，从job这个词的频率可以看出就业问题多么重要。有趣的是，我发现他提到了Youngstown（扬斯敦，美国俄亥俄州东北部城市），而且是2次。

如果想看看词频的频率，可以生成一张表，如下所示：

```
    > head(table(freq))
freq
     2   3   4   5   6  7
    596 354 230 141 137 89

> tail(table(freq))
Freq
    148 157 163 168 174 193
      1   1   1   1   1   1
```

这张表给出了具有某种词频的词的数量，可以看出，有354个词出现了3次，只有1个词出现了193次，就是new。

使用findFreqTerms()函数，找出那些至少出现125次的词：

```
  > findFreqTerms(dtm, 125)
[1] "america" "american" "americans" "jobs" "make" "new"
    [7] "now"     "people"   "work"      "year" "years"
```

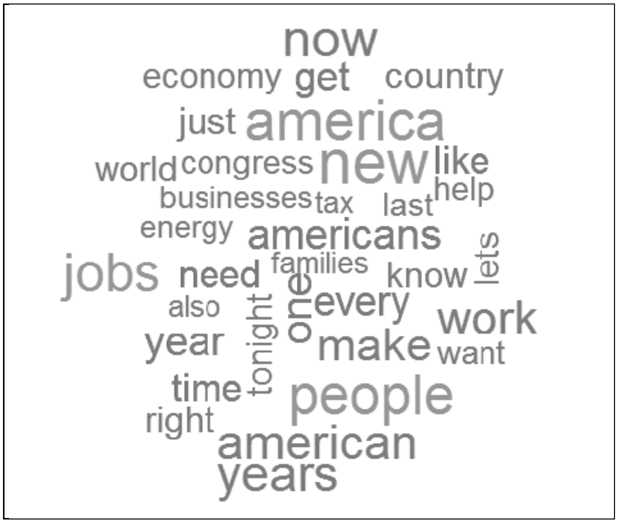
可以通过findAssocs()函数计算相关性，从而找出词与词之间的关联。以job为例，使用0.85作为相关性的下限：

```
> findAssocs(dtm, "jobs", corlimit = 0.85)
$jobs
colleges serve market shouldnt defense  put tax came
    0.97  0.91  0.89     0.88    0.87 0.87 0.87 0.86
```

至于可视化描述，可以生成词云和柱形图。我们生成两种词云，演示不同生成方法。一种通过最小频率法，另一种需要指定词云包含的词的最大数目。第一种使用最小频率的方法，还可以加入代码以指定颜色。词的字号由频率决定，参数scale确定了词的最大字号和最小字号。下面的代码中，最小频率是70：

```
> wordcloud(names(freq), freq, min.freq = 70, scale = c(3, .5),
  colors = brewer.pal(6, "Dark2"))
```

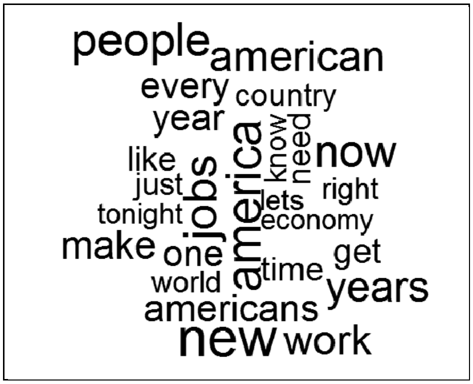
上述命令输出如下。



如果你和我一样不喜欢这些花里胡哨的东西，可以在词云中只显示最频繁的25个词：

```
> wordcloud(names(freq), freq, max.words = 25)
```

上述命令输出如下页图。



要生成柱状图，代码会复杂一些，不论使用的是R基础包、ggplot2还是lattice。以下代码说明了如何使用R基础包为频率最高的10个词生成柱状图：

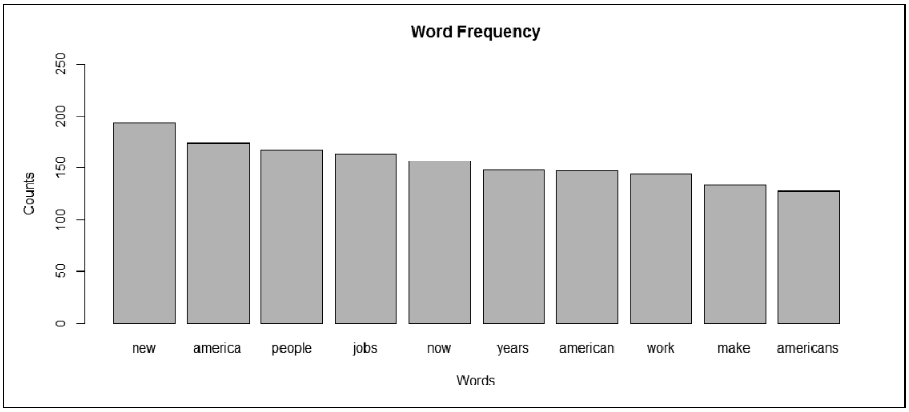
```
> freq <- sort(colSums(as.matrix(dtm)), decreasing = TRUE)

> wf <- data.frame(word = names(freq), freq = freq)

> wf <- wf[1:10, ]

> barplot(wf$freq, names = wf$word, main = "Word Frequency",
 xlab = "Words", ylab = "Counts", ylim = c(0, 250))
```

上述命令输出如下。



下面使用topicmodels包建立主题模型，这个包提供了`LDA()`函数。问题在于要建立多少个主题，看上去3个主题（$k = 3$）是比较符合逻辑的。当然，我建议你也试试其他数量：


```
> library(topicmodels)

> set.seed(123)
```

```
> lda3 <- LDA(dtm, k = 3, method = "Gibbs")

> topics(lda3)
2010 2011 2012 2013 2014 2015 2016
   2    1    1    1    3    3    2
```

可以看到，随着时间的变化，主题的变化也非常有意思。第一篇和最后一篇演讲具有同样的主题分组，看来奥巴马以同样的方式开始和结束了自己的任期。

使用`term()`函数，生成一个对每个主题中的词按词频排序的列表。函数可以指定列表中词的数目，下面看看每个主题中的前25个词：

```
> terms(lda3, 25)
      Topic 1      Topic 2      Topic 3
 [1,] "jobs"       "people"     "america"
 [2,] "now"        "one"        "new"
 [3,] "get"        "work"       "every"
 [4,] "tonight"    "just"       "years"
 [5,] "last"       "year"       "like"
 [6,] "energy"     "know"       "make"
 [7,] "tax"        "economy"    "time"
 [8,] "right"      "americans"  "need"
 [9,] "also"       "businesses" "american"
[10,] "government" "even"       "world"
[11,] "home"       "give"       "help"
[12,] "well"       "many"       "lets"
[13,] "american"   "security"   "want"
[14,] "two"        "better"     "states"
[15,] "congress"   "come"       "first"
[16,] "country"    "still"      "country"
[17,] "reform"     "workers"    "together"
[18,] "must"       "change"     "keep"
[19,] "deficit"    "take"       "back"
[20,] "support"    "health"     "americans"
[21,] "business"   "care"       "way"
[22,] "education"  "families"   "hard"
[23,] "companies"  "made"       "today"
[24,] "million"    "future"     "working"
[25,] "nation"     "small"      "good"
```

主题2包括第一篇和最后一篇演讲。和另外两个主题不同，这个主题真没什么好说的。我们很想知道，下一项分析能否从这两篇演讲中找出点东西来。

主题1包括第一篇演讲之后的3篇演讲，其中能传达信息的词语是jobs、energy、reform和deficit，更不用说关于education的评论，以及前面看到的具有关联的jobs和colleges了。

主题3包括另外两篇演讲。重点似乎确实转移到了经济和商业上，因为提到了security和医疗保健。

下一节会深入分析某一篇演讲的内容，并比较第一篇和最后一篇国会演讲。

13.4.2 其他定量分析

本节将重点放在功能强大的qdap包上。这个软件包可以使用多种数值指标进行文档比较。我们将比较2010年和2016年的演讲。首先需要将文本文件转换为数据框，再进行句法分割。然后将分割出来的句子组合成一个数据框，并建立一个变量以标记演讲年份，用这个变量作为文本分析中的分组变量。处理文本数据时，特别是使用R语言时会比较麻烦。对于这个例子，以下代码在加载数据并进行分析方面似乎效果最好。首先加载qdap程序包，然后使用基础R包中的`readLines()`函数，从文本文件读取数据，并删除不必要的空白字符。我还建议将文本编码设定为ASCII，否则可能出现乱码扰乱分析。可以通过`iconv()`函数完成：

```
> library(qdap)

> speech16 <- paste(readLines("sou2016.txt"), collapse=" ")
Warning message:
In readLines("sou2016.txt") : incomplete final line found on
  'sou2016.txt'

> speech16 <- iconv(speech16, "latin1", "ASCII", "")
```

警告信息不是什么问题，因为它只是告诉我们，文本中最后一行的长度和.txt文件中其他行的长度不一样。现在，可以使用qdap包中的`qprep()`函数了。

这个函数是其他替换函数的打包函数，使用它可以加快预处理过程，但如果需要更多详细分析，那么使用这个函数要小心一些。该函数打包的函数如下所示。

- `bracketX()`：去掉括号
- `replace_abbreviation()`：替换缩略语
- `replace_number()`：将数字替换为单词，例如，可以将100替换为one hundred
- `replace_symbol()`：将符号替换为单词，例如，可以将@替换为at

```
> prep16 <- qprep(speech16)
```

我们要做的一些其他预处理工作包括替换缩写（can't变为cannot）；剔除停用词，在这个例子中，剔除前100个常用的停用词；剔除不需要的字符，除了句号和问号。很快你就能知道这些操作的作用了：

```
> prep16 <- replace_contraction(prep16)

> prep16 <- rm_stopwords(prep16, Top100Words, separate = F)

> prep16 <- strip(prep16, char.keep = c("?", "."))
```


下面就是这个分析中最重要的部分，将文档分割成句子，并加上要作为分组变量的演讲年份。这个过程还会创建`tot`变量，tot的意思是谈话的顺序（Turn of Talk），作为表示句子顺序的指标。这个指标在你分析对话时特别有用，比如辩论和问答场景：

```
> sent16 <- data.frame(speech = prep16)

> sent16 <- sentSplit(sent16, "speech")

> sent16$year <- "2016"
```

对2010年的演讲重复以上步骤：

```
> speech10 <- paste(readLines("sou2010.txt"), collapse=" ")

> speech10 <- iconv(speech10, "latin1", "ASCII", "")

> speech10 <- gsub("(Applause.)", "", speech10)

> prep10 <- qprep(speech10)

> prep10 <- replace_contraction(prep10)

> prep10 <- rm_stopwords(prep10, Top100Words, separate = F)

> prep10 <- strip(prep10, char.keep = c("?", "."))

> sent10 <- data.frame(speech = prep10)
> sent10 <- sentSplit(sent10, "speech")

> sent10$year <- "2010"
```

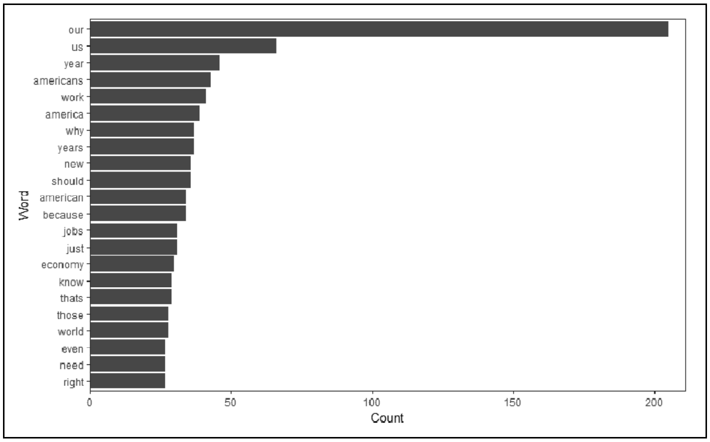
将这两个独立年份的数据合成一个数据框：

```
> sentences <- data.frame(rbind(sent10, sent16))
```

qdap包的一大优点就是，可以非常方便地完成文本探索工作。和以前做的一样，看看名词的频率图：

```
> plot(freq_terms(sentences$speech))
```

上述命令输出如下。



可以建立一个词频矩阵，表示每篇演讲中每个单词的数量：

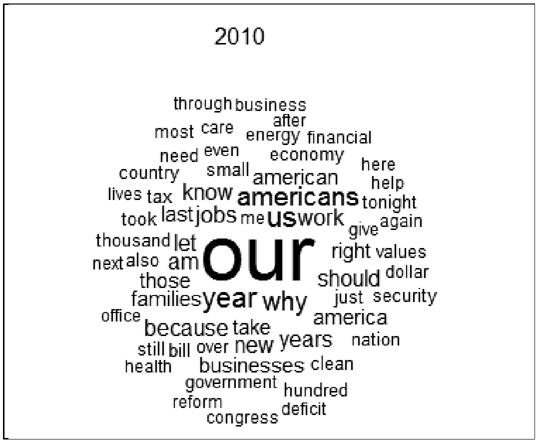
```
> wordMat <- wfm(sentences$speech, sentences$year)

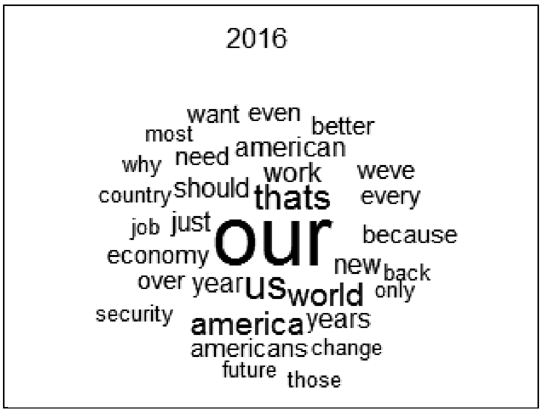
> head(wordMat[order(wordMat[, 1], wordMat[, 2],decreasing =
  TRUE),])
          2010 2016
our        120   85
us          33   33
year        29   17
americans   28   15
why         27   10
jobs        23    8
```

这个矩阵还可以转换为文档–词矩阵，如果你有这个需求的话，可以使用函数`as.dtm()`。下一步使用qdap中的功能，为每年建立一个词云：

```
> trans_cloud(sentences$speech, sentences$year, min.freq = 10)
```

上述命令会生成以下两幅图。



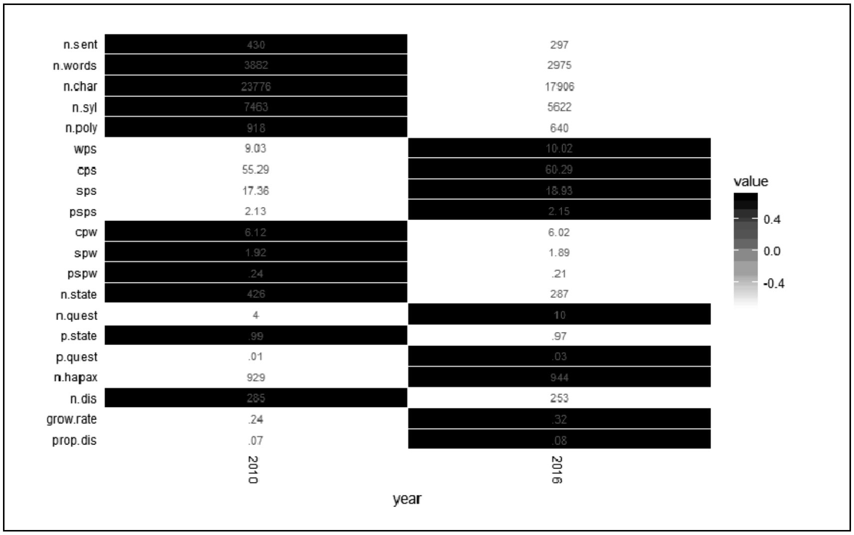


还可以对文档中的词进行综合统计。下图由软件包中的综合统计绘制而成。这个图因为只有两篇文档，所以视觉效果打了折扣。尽管如此，它依然能揭示一些信息。可以通过?word_stats命令看到对这个统计的完整解释：

```
> ws <- word_stats(sentences$speech, sentences$year, rm.incomplete = T)

> plot(ws, label = T, lab.digits = 2)
```

上述命令输出如下。



请注意，2016年的演讲要比2010年的短很多，少100多个句子和差不多1000个单词。还有，2016年（10个问句）相对于2010年（4个问句），更多地使用了问句这种修辞手法。

如果要比较极性（情感评分），可以使用polarity()函数，指定文本和分组变量即可：

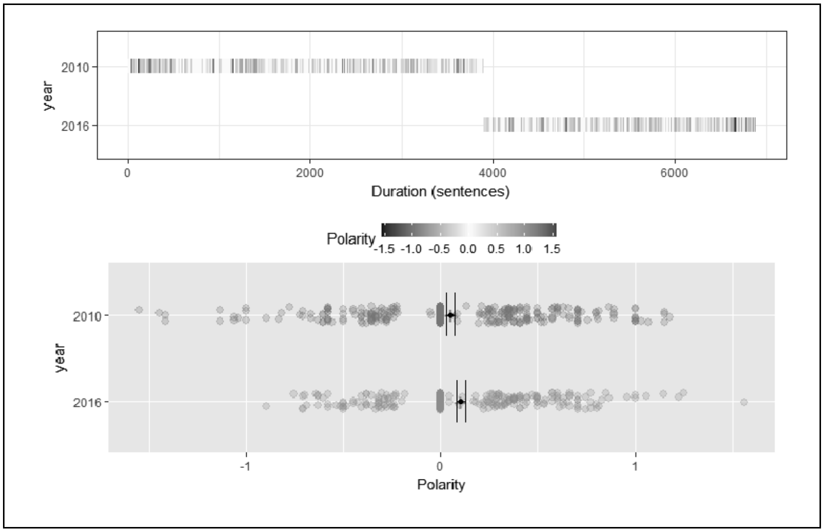
```
    > pol = polarity(sentences$speech, sentences$year)

> pol
     year total.sentences total.words ave.polarity sd.polarity
       stan.mean.polarity
1 2010                 435         3900        0.052       0.432
   0.121
2 2016                 299         2982        0.105       0.395
   0.267
```

stan.mean.polarity表示标准化后的极性均值，就是用极性均值除以标准差。可以看到2016年的标准化极性均值（0.267）要比2010年（0.121）稍高一些。这也符合我们的预期，总统先生想以更积极的态度结束任期。你也可以通过统计图表示数据。统计图中有两幅图，第一幅表示随时间变化的句子极性，第二幅表示极性的分布：

```
> plot(pol)
```

上述命令输出如下。



这张统计图有点难懂，我尽力解释一下。2010年的演讲以非常悲观的情感开始，而且整体上比2016年悲观。通过pol对象中的数据框，可以找出最悲观的句子。先找出句子的标号，然后引用它：

```
    > pol.df <- pol$all

> which.min(pol.df$polarity)
[1] 12

> pol.df$text.var[12]

[1] "One year ago, I took office amid two wars, an economy rocked
      by a severe recession, a financial system on the verge of
         collapse, and a government deeply in debt.
```

这就是悲观情感！具有讽刺意味的是，今天的政府更加负债累累。下面看看易读性指数：

```
    > ari <- automated_readability_index(sentences$speech,
      sentences$year)

> ari$Readability
  year word.count sentence.count character.count
1 2010       3900            435           23859
2 2016       2982            299           17957
  Automated_Readability_Index
1                    11.86709
2                    11.91929
```

不出意外，两篇演讲的指数基本相同。下面看看正式度的分析。用R进行正式度分析需要几分钟的时间：

```
> form <- formality(sentences$speech, sentences$year)
```

```
> form
  year word.count formality
1 2016       2983     65.61
2 2010       3900     63.88
```

看上去非常接近。可以检查演讲中各种词的比例并绘制统计图，但对分析没什么帮助。这个例子中的词的比例如下：

```
> form$form.prop.by
  year word.count  noun   adj  prep articles pronoun
1 2010       3900 44.18 15.95  3.67        0       4.51
2 2016       2982 43.46 17.37  4.49        0       4.96
  verb adverb interj other
1 23.49  7.77   0.05  0.38
2 21.73  7.41   0.00  0.57
```

现在计算多样性指标。两年的结果还是几乎相同。可以做一张统计图（`plot(div)`），但是因为太接近了，所以还是没有什么意义。值得一提的是，奥巴马2010年国会演讲的作者是乔·费夫洛，2016年的作者则是科迪·基南：

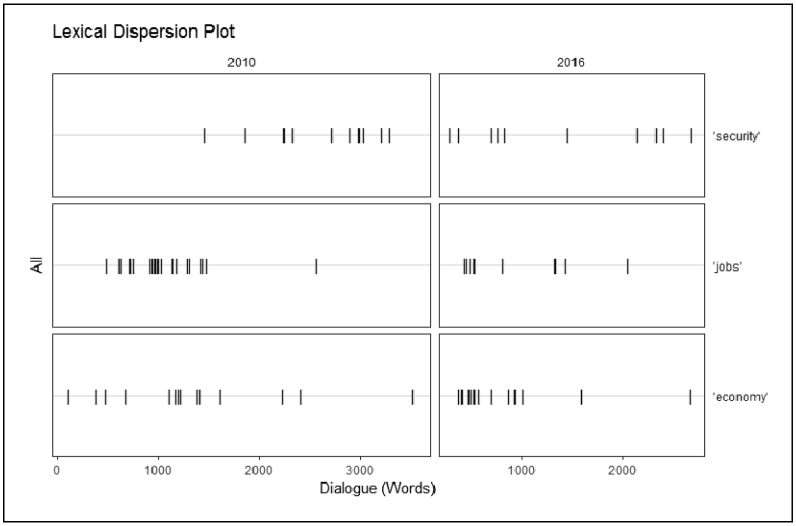
```
> div <- diversity(sentences$speech, sentences$year)
```

```
> div
  year   wc simpson shannon collision berger_parker brillouin
1 2010 3900   0.998   6.825     5.970         0.031     6.326
2 2015 2982   0.998   6.824     6.008         0.029     6.248
```

分散度图是我最喜欢的统计图之一，它展示了词在整篇文本中的分散程度。我们看一下security、jobs和economy的分散度：

```
> dispersion_plot(sentences$speech,
  rm.vars = sentences$year,
  c("security", "jobs", "economy"),
  color = "black", bg.color = "white")
```

上述命令输出如下页图。



这个图非常有意思，因为你可以形象地表示2010年的演讲有多长。2010年演讲的前半部分特别强调工作机会，而在2016年，演讲的前半部分似乎更注重对整体经济的论述。毫无疑问，是关于奥巴马自己采取了多少措施将经济从崩溃的边缘挽救回来的。2010年，国家安全问题直到演讲的后半部分才提到，而在最后的演讲中，安全问题贯穿始终。这样你就可以知道并理解文本分析如何揭示深刻的知识，包括一个人的想法和他的优先级，而且明白如何与其沟通。

这样就完成了对两篇演讲的分析。必须承认，这些演讲我一次也没有听过。实际上，从里根总统开始，除了2002年，我没有收看过任何一次国情咨文。这项分析使我知道，随着时间的推移，国会演讲的主题和形式如何改变，以适应政治需要。同时由于演讲的正式性，总体风格和句法结构是保持一致的。请记住，本章的代码适合于几十种（如果没有几百种的话）文档或言论的文本挖掘，而且文档可以有多个作者，例如电影剧本、法律程序、采访记录、社交媒体等。的确，文本挖掘可以将现存的定性混乱改变为定量秩序。

13.5　小结

本章研究了如何使用文本挖掘方法来处理大规模文本数据。我们介绍了文本挖掘的实用框架，包括数据准备、词频分析和可视化，以及通过tm包使用LDA建立主题模型。框架还包含其他定量分析技术，例如极性分析和正式度，这可以让我们更深入地理解文本中的词汇，也可以说是风格。通过qdap包可以实现这些分析。我们应用这个框架分析了奥巴马总统的7年国会演讲，结果发现，尽管这些演讲基本上都是一个风格，但因为政治形势的变化，其关键信息也随着时间的推移而发生了变化。虽然很难介绍所有文本挖掘技术，但本章内容对于我们将要面临的多数问题都适用。下一章将从建模工作中解脱出来，重点介绍如何在云上运行R。这可以扩展你的机器学习能力，满足你解决任何问题的需要。

第 14 章

在云上使用R语言

“如果有人问我什么是云计算，那我一时半会儿还真说不清楚。我会告诉他们，简单地说，云计算就是一种运行业务的更好方式。”

——马克·贝尼奥夫，云计算软件服务提供商赛富时（Salesforce.com）CEO

因为我不是一个想从云计算中获取利润的公司CEO，所以还是给云计算下个定义。我很喜欢微软公司的这个定义（参见https://azure.microsoft.com/en-us/overview/what-is-cloud-computing）：

> 简单地说，云计算就是通过互联网（云）提供的计算服务——服务器、存储、数据库、网络、软件、分析，等等。提供这些计算服务的公司称为云供应商，通常根据使用情况对云计算服务进行收费，这和你为家里的水电付费非常相似。

如果你还没有使用云服务进行机器学习，那么我可以保证在不远的将来，你会使用的。我知道有些人会担心数据失控、安全问题等，一个创业公司的CEO就问过我这样的问题。我会反问有这种担心的人：“如果你们认为保存在笔记本电脑上的数据是安全的，那么会通过Wi-Fi来访问这些数据吗？”如果答案是“会”，那么这就是在使用云服务，与上面所说的云服务的区别只是存储硬件的环境不一样。

就是这样。你想把办公室装满服务器，弄得像个地牢一样？还是想让别人通过他们安全的、有冗余备份的和遍布全球的基础设施来解决问题呢？

基于云计算使用R语言可以在多个工作地点之间达成无缝连接,也可以获得极大的计算能力，而且可以按照需要快速向上或向下扩展。云计算可以节省大量成本。

有很多种方式可以在云上使用R，我要使用的是亚马逊的云服务平台AWS（Amazon Web Services）和他们的弹性计算云（Elastic Compute Cloud，EC2）。我使用亚马逊云服务进行说明是因为，它是我使用的第一个云服务，而且我已经非常熟练了。这并不是说我认为它比其他产品要好。我现在不这样认为，将来也不会，除非杰夫·贝佐斯（亚马逊集团董事会主席兼CEO）选择我去执行一次载人航天任务，我的态度才会改变。

无论如何，本章的目的是带领并教会你在不写一行Linux代码的情况下，快速地在云上使用R语言和Rstudio。为了最大限度地利用AWS的能力和它花样繁多的在线工具，你可以学习Linux代码，并通过**SSH**（Secure Shell，安全外壳协议）来使用。对于本章的内容，我们要创建并启动一个名为**实例**（instance）的虚拟机，然后通过网页浏览器登录Rstudio，并使用其中的一些功能。网上有很多这样的教程，但我的目标是使你以最简单和最快速的方式开始，并且**今天**就可以在云上使用R。

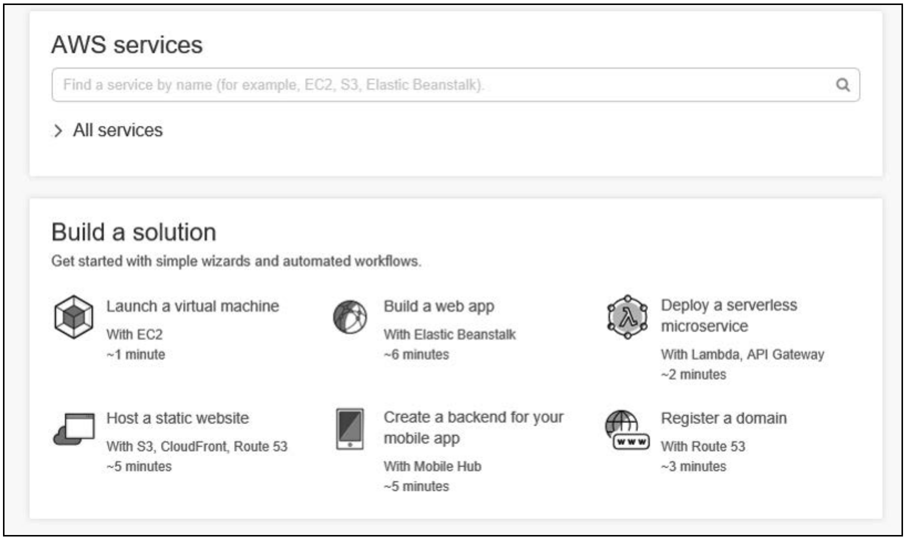
14.1　创建 AWS 账户

第一件事就是要注册一个AWS账户：

https://aws.amzon.com/

这是这个练习的唯一前提条件。注册过程需要一个信用卡，但是你在本章中要做的练习不会花一分钱，因为我们使用的是免费实例。然后你就可以快速启动一个新的实例，这个实例具有你需要的强大计算能力，还可以在完成练习后停止或结束。当你创建账户并登录时，可以选择是否建立安全组。创建实例时，我会通过建立一个新的安全组来说明它的用处。安全组可以让你控制谁可以访问实例，以及如何访问实例。还有，现在先不要创建**密钥对**，除非你特别需要，否则可以在以后创建。

这些步骤都完成之后，登录进入你的AWS控制台，如下所示。

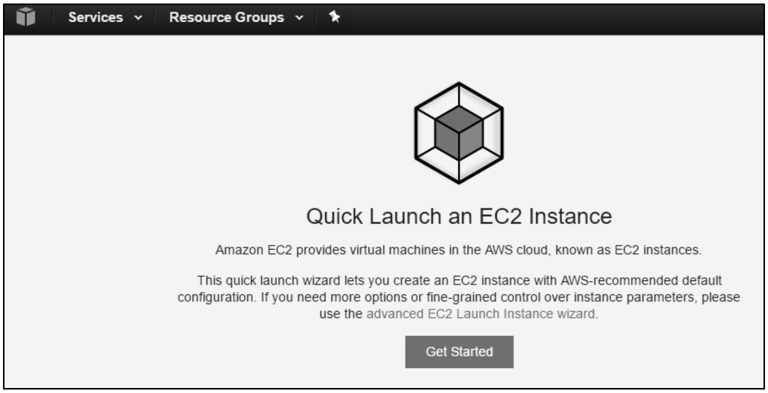


点击智能超链接**Launch a virtual machine**即可创建并启动一个虚拟机。

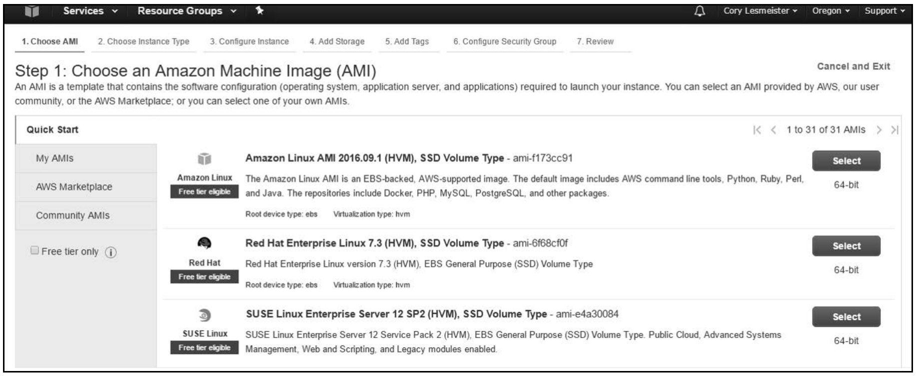
14

14.1.1　启动虚拟机

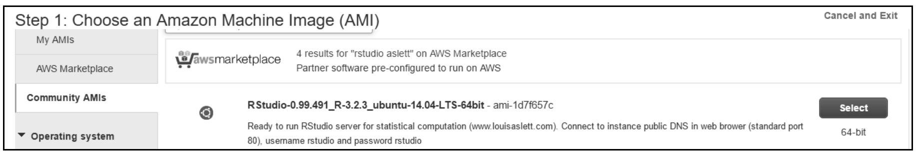
启动虚拟机的超链接会带你进入如下页面。



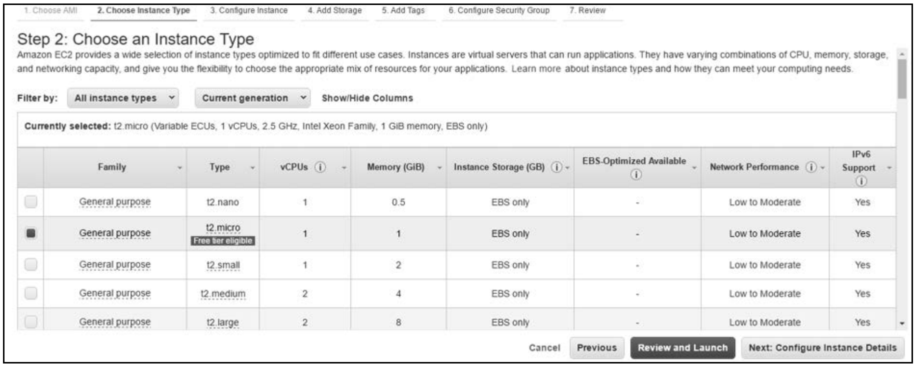
不要点击**Get Started**按钮，而是点击**advanced EC2 Launch Instance Wizard**，如下所示。



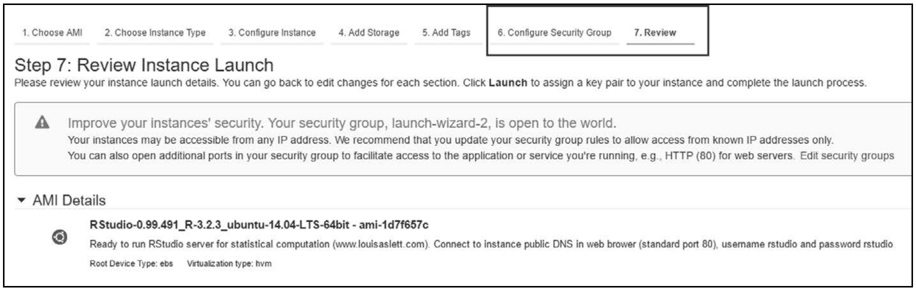
如果你已经有了一些经验，就可以使用不同的**亚马逊机器镜像**（Amazon Machine Image，AMI），并可以定制在AWS上使用R的方式。但本章目标是快速而又简单地在云上使用R，考虑到这一点，AWS用户建立了一些社区AMI，其中已经包含了R和Rstudio。所以，可以在**Quick Start**之后点击**Community AMIs**，这时会弹出一个搜索框，我建议先使用由Louis Aslett维护的AMI：http://www.louisaslett.com/RStudio_AMI/。搜索rstudio aslett可以找到这个AMI，会显示下面的网页，点击Select按钮，如下所示。



这样就到了第二步——选择实例类型。我选择的是t2.micro免费层实例。



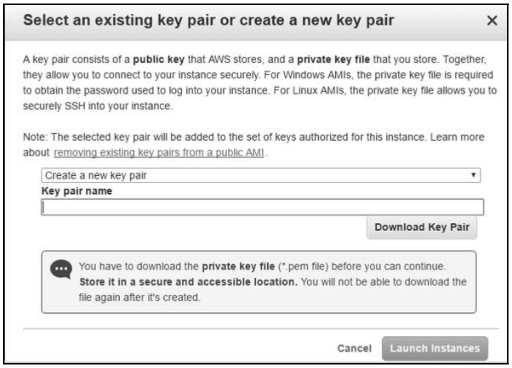
如果已经选择了需要的实例类型，则可以点击Review and Launch。因为这是一个已有的AMI，所以可以跳过第七步——Review标签页。此时可以启动实例，但我们点击一下**第六步——Configur Security Group**。



在启动过程的这一步，你可以建立一个安全组，也可以使用一个现有的。下面是创建**新安全组**的示例。



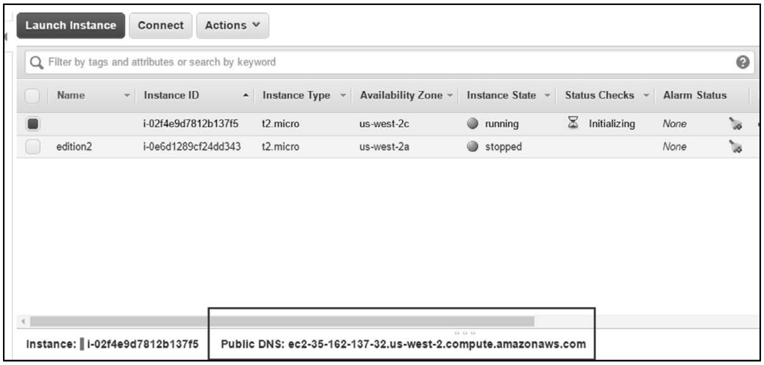
完成该步骤后（可以完全不做修改），点击**Review and Launch**。这会带你回到**第七步**，此时可以点击**Launch**，来到选择新密钥对或现有密钥对的页面。



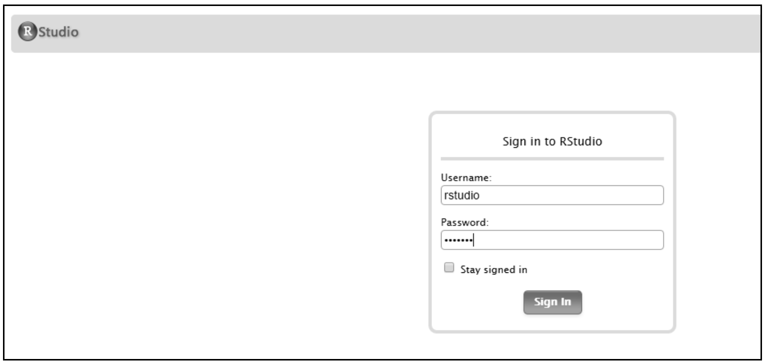
完成之后，点击**Launch Instance**，回到AWS控制台。

14.1.2　启动 Rstudio

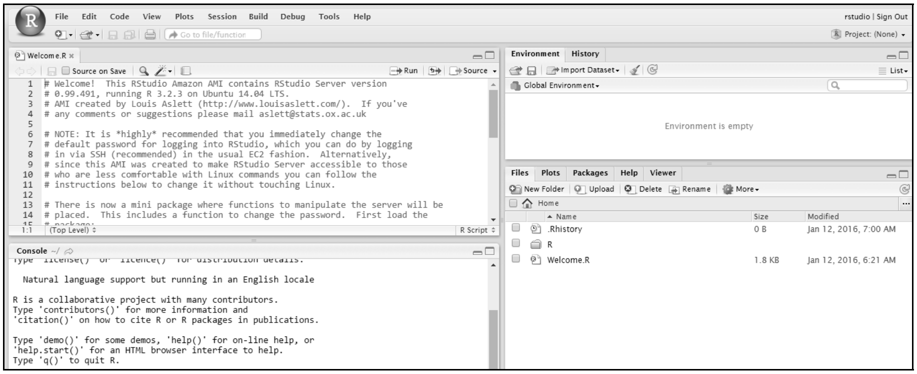
启动实例之后，回到AWS控制台并选择该实例时，你会看到如下页面。



注意选中实例的**Public DNS**。要在你的网页浏览器中启动Rstudio，有这个就足够了。要在浏览器中启动Rstudio，需要先来到Rstudio的登录页面，用户名和密码都是rstudio。



就是这样！你已经在一个虚拟机上运行Rstudio了。它看上去应该如下所示。



在左上角的**Source Panel**窗格中，有关于如何修改密码和连接到Dropbox的指示。

为了演示如何从网络上加载数据，我要从github上加载一个前面章节中使用过的.csv文件。试试climate.csv好吗？首先要安装和加载RCurl包：

```
> install.packages("RCurl")

> library(RCurl)
```

然后，需要获得GitHub上的数据连接：

```
> url <-
"https://raw.githubusercontent.com/datameister66/data/master/climate.csv"
```

然后，将数据读取到Rstudio中：

```
> climate <-read.csv(text = getURL(url))
```

确定读取数据的结果：

```
> head(climate)
  Year CO2   Temp
1 1919 806 -0.272
2 1920 932 -0.241
3 1921 803 -0.187
4 1922 845 -0.301
5 1923 970 -0.272
6 1924 963 -0.292
```

就是这样。现在，你已经成为一个基于云计算的机器学习勇士了，你几乎可以像使用自己的计算机一样在虚拟机上进行各种操作。

请注意，如果你完成操作并退出Rstudio，请一定回到控制台停止实例。

14.2　小结

在本书最后一章中，我们介绍了如何快速而又简单地在云上运行R语言和Rstudio。在这个练习中，我们通过使用AWS，循序渐进地介绍了如何在云上创建虚拟机（实例）、配置虚拟机、启动虚拟机，以及在浏览器中运行Rstudio。最后，通过从GitHub上读取climate.csv文件，说明了从网络上加载数据有多么容易。通过本章对云计算的简单介绍，你可以在任何有互联网连接的地方开展工作，并且可以对实例进行快速扩展和收缩，以满足你的需求。本书的主要章节到此结束。我希望你能喜欢本书内容，并能将书中介绍的方法和你逐渐学会的其他方法在实际中加以应用。谢谢！

附　录

R语言基础

"我曾经一下子废掉了1000行代码，那是我最有成效的一天。"

——肯・汤普森

本部分介绍R语言的基本编程语法和常用功能。希望可以带你进入R语言的世界，激发你对R语言的学习兴趣。我们的目标如下：

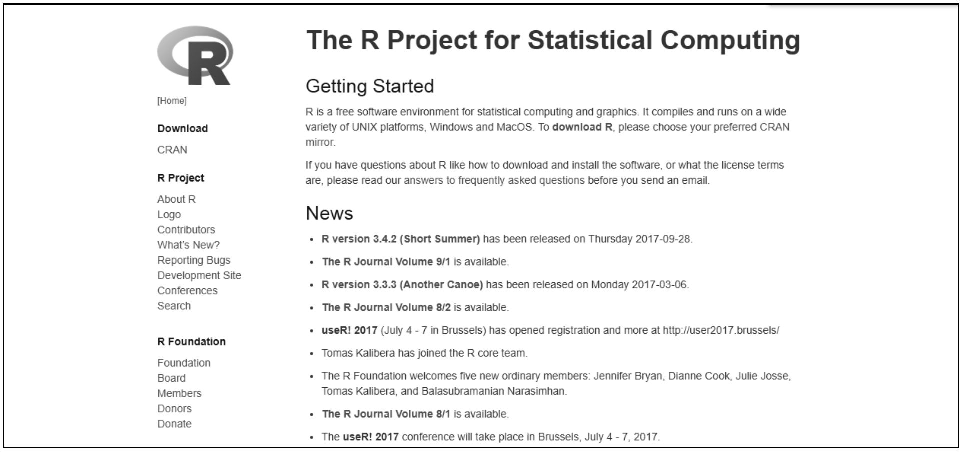
- 安装R语言与RStudio
- 建立并使用向量
- 建立数据框与矩阵
- 了解数学与统计函数
- 生成简单的统计图
- 介绍数据处理扩展包dplyr
- 安装与加载R程序包

附录中的每个示例都在前面的章节中介绍过。如果你没有使用过R，那么这是一个非常好的开始，本部分可以使你更好地理解前面各章内容。

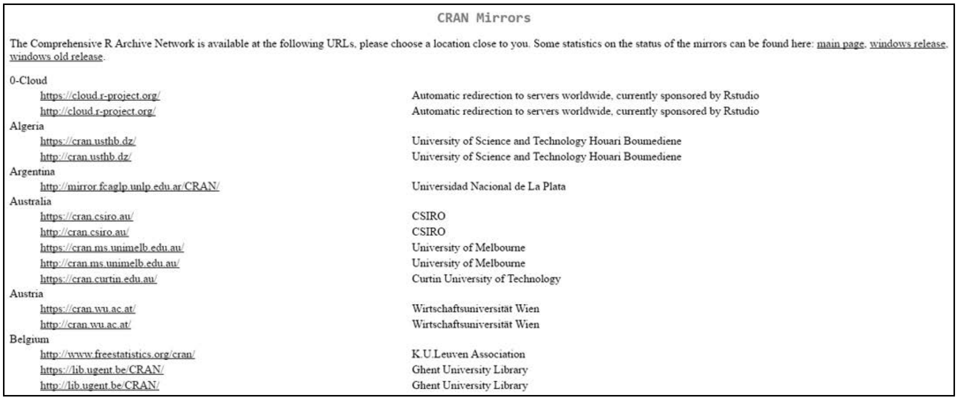
1. 安装与运行R

本节需要完成两个任务：第一，安装最新版本的R语言；第二，安装RStudio，即R语言的**集成开发环境**（Integrated Development Environment，IDE）。

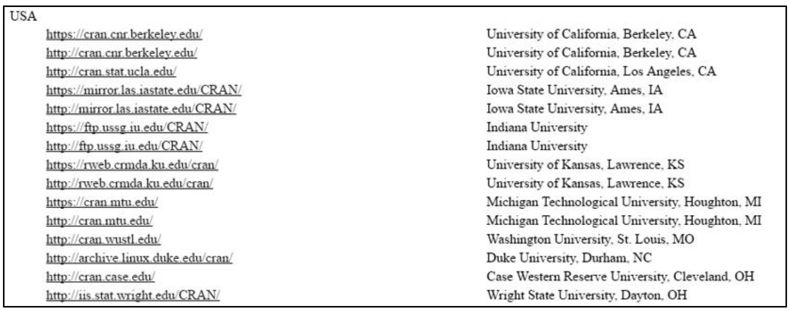
首先查看R的主页，地址为https://www.r-project.org/，类似下页屏幕截图。



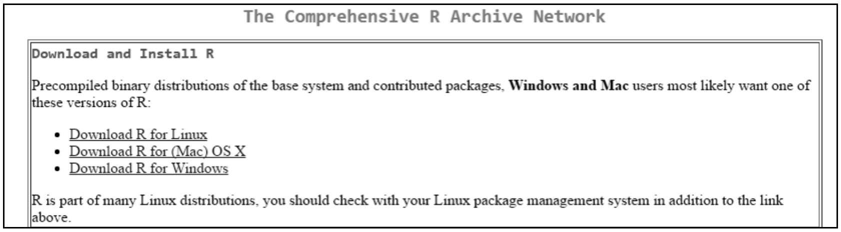
可以看到**download R**链接，在**News**部分可以看到R的最新版本为3.4.2。现在，点击一个链接，**Download**下面的**CRAN**或者**Getting Started**下面的**download R**都可以，然后进入如下页面，即**CRAN镜像**。



下面的链接是按照国家和字母顺序排列的，可以将你带入下载页面。因为我在美国，所以向下滚动页面，可以发现有很多可用的链接。

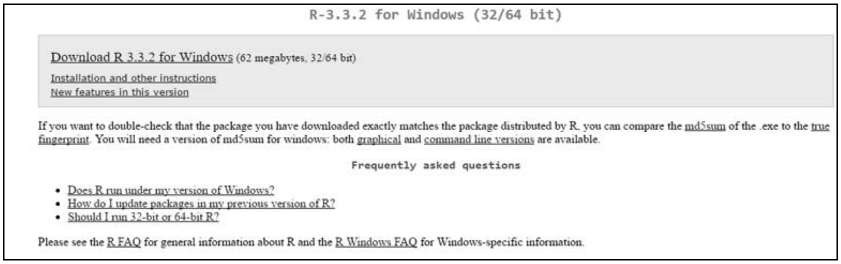


如果你找到了和你所在地点相似的链接，请点击，然后会看到类似下图的页面。

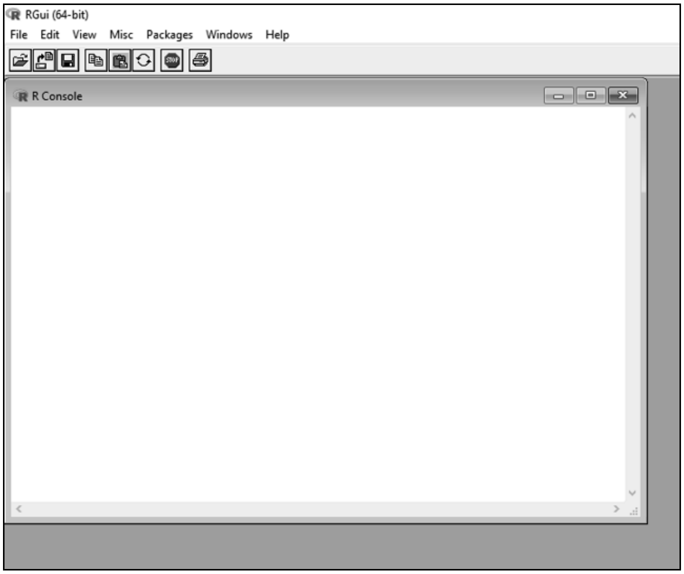


现在，点击并选择你的操作系统。

我们现在是首次安装R，所以点击**install R for the first time**，然后来到如下页面，开始下载安装程序。



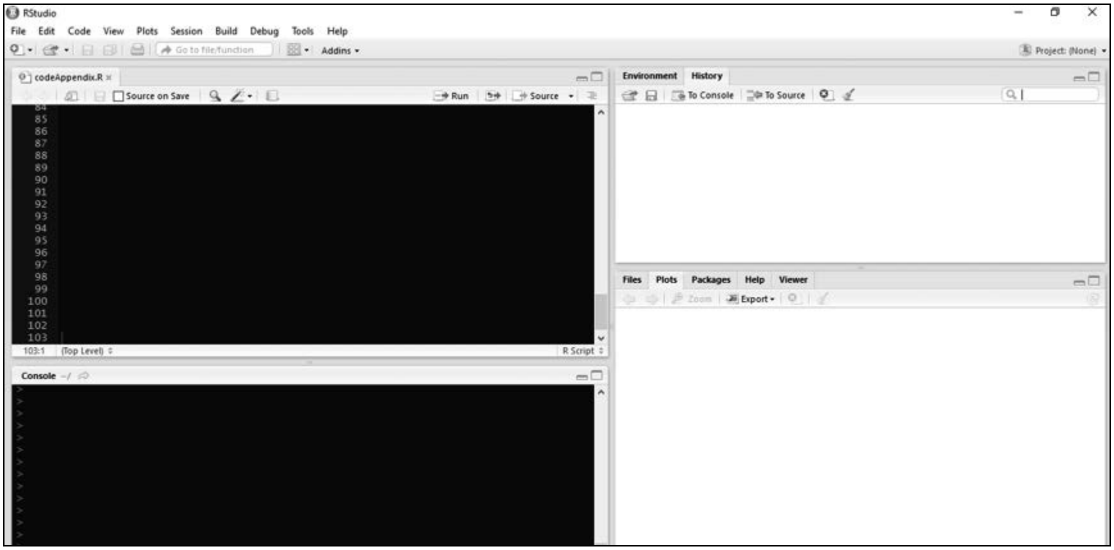
现在，你就可以像安装其他程序一样安装R了。安装完成之后运行R，你会看到基础的**图形用户界面**。



现在，你可以运行本书中的所有代码了。但是，我们强烈推荐在RStudio的IDE环境下使用R（有免费版本）。可以通过下面的链接下载免费的RStudio：

https://www.rstudio.com/products/RStudio/

在这个页面，你可以找到免费版本和商业版本的下载。不用说，我们直奔免费版本下载并安装。程序安装完毕，第一次打开之后，你可以看到如下界面。请记住，如果加载的程序包和操作系统不同，界面也会有些差别。

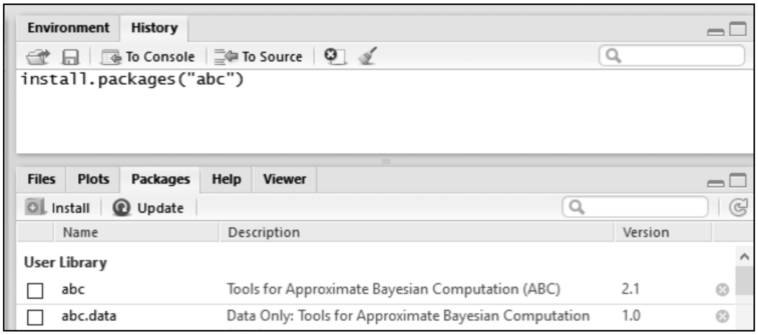


请注意，左边是带有命令行提示符的控制台，这和前面的图是一样的。IDE可以提高用户体验，比如你可以管理**Environment**和**History**（右上），以及**Files**、**Plots**、**Packages**和**Help**（右下）。

关于RStudio的功能有很详细的教程，不用在这上面花费过多精力，直接查看一些重要的功能。R的最大优点之一就是，拥有数量庞大的高质量软件包，可以进行各种分析。下面看看如何使用IDE加载一个软件包，比如abc包，它用来进行近似贝叶斯计算。使用命令行提示符，输入如下命令：

```
> install.packages("abc")
```

命令运行之后，请注意右下角的窗格（请确保点击选中**Packages**标签页）中，abc包和它的依赖包abc.data一起安装完成。现在，点击右上角**History**标签页，可以看到运行过的加载软件包的命令。



现在，点击**To Console**按钮，这个命令就会被放到命令行提示符前面。点击**To Source**按钮会看到一个新区域被打开，供你组织项目脚本。

命令`install.packages()`从历史操作进入源文件。如果你测试过代码，获得了满意的效果，那么可以把代码放在源文件中。你可以将其保存到硬盘，或者用邮件发送给别人。本书中的每章代码都保存在一个源文件中。

2. 使用R

系统一切就绪之后，我们开始第一个命令。R支持引号中的字符串和简单数字。下面使用一个字符串命令和一个数字命令，此处的输出和输入是一样的：

```
    > "Let's Go Sioux!"
[1] "Let's Go Sioux!"

> 15
[1] 15
```

R还可以用作计算器：

```
    > ((22+5)/9)*2
[1] 6
```

R的亮点始于能够建立向量。下面将斐波那契数列中的前10个数字组成一个向量，使用`c()`函数，它表示将几个值组合成向量或列表（c是concatenate的首字母，表示连接）：

```
   > c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34) #Fibonacci sequence
[1]  0 1 1 2 3 5 8 13 21 34
```

请注意，在上面的代码中，我添加了注释`Fibonacci sequence`。在R的命令行中，关键字`#`后面的内容是不执行的。

下面建立一个对象，使之包含数列中的这些数字。可以将任何向量或列表分配给一个对象。在多数R代码中，你会看到赋值符号`<-`，读作“赋值为”。建立一个对象`x`，保存斐波那契数列：

```
> x <- c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
```

如果想看看对象x中的内容，在命令行中输入x即可：

```
> x
[1] 0 1 1 2 3 5 8 13 21 34
```

你可以在对象后面加上中括号来选择向量的子集。下面的代码可以让你得到序列的前3个观测：

```
> x[1:3]
[1] 0 1 1
```

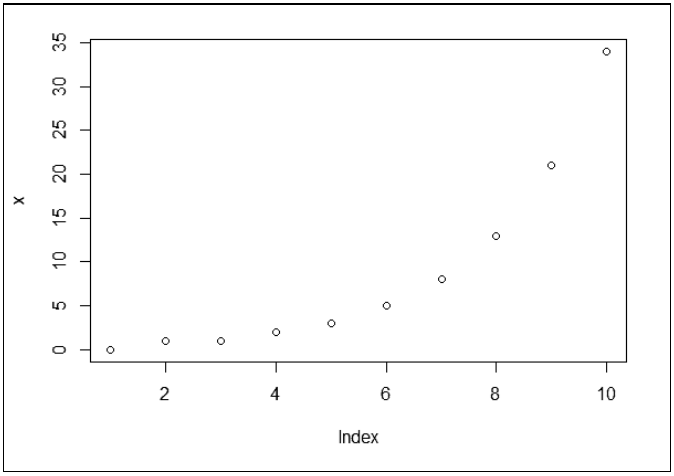
可以在中括号内的数字前加上一个负号，以将该数字对应的观测排除在外：

```
> x[-5:-6]
[1]  0  1  1  2  8 13 21 34
```

使用plot()函数，将序列可视化：

```
> plot(x)
```

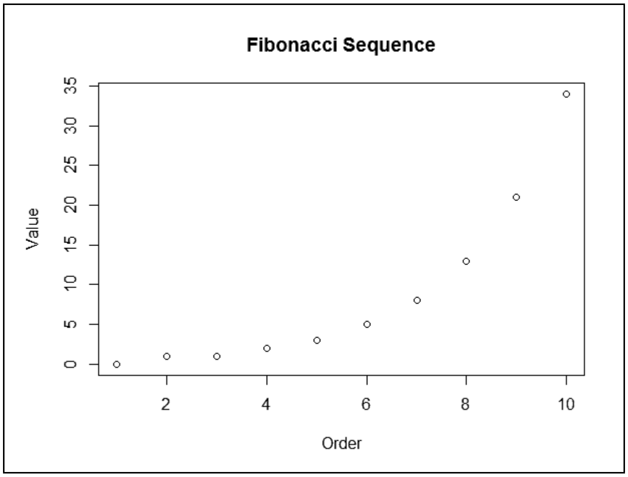
上述命令输出如下。



使用main = ...、xlab = ...和ylab = ...，可以轻松地为统计图加上标题和坐标轴标签：

```
> plot(x, main = "Fibonacci Sequence", xlab = "Order", ylab = "Value")
```

上述命令输出如下页图。



R中有很多函数可以对向量进行转换。下面建立一个新对象y，表示x的平方根：

```
> y <- sqrt(x)
```

```
> y
[1] 0.000000 1.000000 1.000000 1.414214 1.732051 2.236068 2.828427
[8] 3.605551 4.582576 5.830952
```

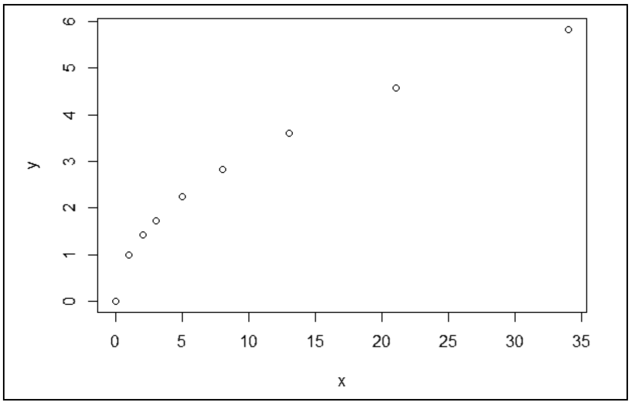
需要重点提示，当你不确定函数的语法时，可以在函数前面加一个?，这样即可调用相关的帮助信息。试一下吧！

```
> ?sqrt
```

这样就打开了这个函数的帮助信息。建立x和y之后，可以生成一张散点图：

```
> plot(x, y)
```

上述命令输出如下。



下面建立一个常数对象，然后将这个对象作为标量，乘以向量x，建立一个新的向量x2：

```
> z <- 3

> x2 <- x * z

> x2
 [1]  0  3  3  6  9 15 24 39 63 102
```

R也支持逻辑运算。例如，我们可以测试一个值是否小于另一个值：

```
> 5 < 6
[1] TRUE

> 6 < 5
[1] FALSE
```

第一个例子中，R返回TRUE，第二个例子返回FALSE。如果你想知道一个值是否等于另一个值，那么应该使用两个等号（测试两个值是否相等）。请注意，一个等号表示赋值，不能测试两个值是否相等。在下面的例子中，我们可以知道刚建立的斐波那契数列中的值是否为0：

```
> x == 0
[1] TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

输出是一个列表，可以清楚地看到，x向量的第一个值确实为0。总结一下，R的关系运算符包括<=、<、==、>、>=、!=，分别表示小于等于、小于、等于、大于、大于等于、不等于。

还要介绍函数rep()和seq()，这两个函数在建立向量时非常有用。例如，rep(5,3)会将5重复3次，对字符串也同样有效：

```
> rep("North Dakota Hockey, 2016 NCAA Division 1 Champions", times=3)
[1] "North Dakota Hockey, 2016 NCAA Division 1 Champions"
[2] "North Dakota Hockey, 2016 NCAA Division 1 Champions"
[3] "North Dakota Hockey, 2016 NCAA Division 1 Champions"
```

下面演示seq()的用法。假设想建立一个数列，从0到10，每次增加2，代码如下所示：

```
> seq(0, 10, by = 2)
[1]  0 2 4 6 8 10
```

3. 数据框与矩阵

下面建立数据框，它是变量（向量）的集合。先建立一个向量(1, 2, 3)，再建立一个向量(1, 1.5, 2.0)。然后使用rbind()函数，将两个向量按行合并：

```
> p <- seq(1:3)

> p
[1] 1 2 3
```

```
> q = seq(1, 2, by = 0.5)

> q
[1] 1.0 1.5 2.0

> r <- rbind(p, q)

> r
  [,1] [,2] [,3]
p    1  2.0    3
q    1  1.5    2
```

结果是一个两行列表，每行有3个值。可以使用`str()`函数确定数据结构，在本例中，可以看到有两个列表，一个是`p`，一个是`q`：

```
    > str(r)
num [1:2, 1:3] 1 1 2 1.5 3 2
- attr(*, "dimnames")=List of 2
 ..$ : chr [1:2] "p" "q"
 ..$ : NULL
```

下面，使用`cbind()`函数将两个向量按列合并：

```
    > s <- cbind(p, q)

> s
     p   q
[1,] 1 1.0
[2,] 2 1.5
[3,] 3 2.0
```

要将这个结果转换为数据框，可以使用`data.frame()`函数。转换完成之后，检查数据结构：

```
    > s <- data.frame(s)

> str(s)
'data.frame':3 obs. of 2 variables:
 $ p: num  1 2 3
 $ q: num  1 1.5 2
```

这样就建立了一个数据框`s`，它有两个变量，每个变量有3个观测。可以使用`names()`函数修改变量名称：

```
    > names(s) <- c("column 1", "column 2")

> s
  column 1 column 2
1        1      1.0
2        2      1.5
3        3      2.0
```

尝试使用`as.matrix()`函数，将数据框转换为矩阵。有些R包要求使用数据框进行分析，但

有些R包要求使用矩阵。可以根据需要在数据框和矩阵之间随意转换：

```
    > t <- as.matrix(s)

> t
     column 1 column 2
[1,]        1      1.0
[2,]        2      1.5
[3,]        3      2.0
```

可以查看矩阵和数据框中的值。例如，我们想知道第一个变量中第一个观测的值。对于本例，只需在中括号中指定第一行和第一列，如下所示：

```
    > t[1,1]
column 1
       1
```

假如想得到第二个变量（列）中的所有值，只要将行的位置留空即可。但别忘了，应当在所需的列前面加一个逗号：

```
    > t[,2]
[1] 1.0 1.5 2.0
```

与之相反，如果只想看看前两行中的值，那么对于本例，只需使用一个冒号：

```
    > t[1:2,]
     column 1 column 2
[1,]        1      1.0
[2,]        2      1.5
```

假如有一个10个变量和100个观测的数据框或矩阵，想建立一个子集，包括前70个观测和第1、3、7、8、9、10个变量。应该怎么操作呢？

使用冒号、逗号、连接函数和中括号即可轻松完成，如下所示：

```
> new <- old[1:70, c(1,3,7:10)]
```

你会注意到自己可以轻松操作想要的观测和变量。还可以轻松排除变量。假设只想排除第一个变量，可以在第一个变量前面加个负号，如下所示：

```
> new <- old[, -1]
```

R的基本数据处理功能非常强大，本书的主体章节介绍了更多高级数据处理技术。

4. 创建摘要统计量

本节介绍几种基本的统计函数，用来测量数据的集中趋势和分散性，并制作简单的统计图。我们要先清楚一个问题：R在计算中如何处理缺失值？为了说明这个问题，先建立一个带有缺失值（R语言用`NA`表示缺失值）的向量，然后使用`sum()`函数将向量的所有分量值相加求和：

```
> a <- c(1, 2, 3, NA)

> sum(a)
[1] NA
```

在SAS中，直接将非缺失值相加。但和SAS的处理方式不同，R不加总非缺失值，而是直接返回NA，表示至少有一个缺失值。可以建立一个新的删除了缺失值的向量，也可以用参数na.rm = TURE表示在计算时排除缺失值：

```
> sum(a, na.rm = TRUE)
[1] 6
```

以下函数可以测量向量中数值的集中趋势和分散程度：

```
> data <- c(4, 3, 2, 5.5, 7.8, 9, 14, 20)

> mean(data)
[1] 8.1625

> median(data)
[1] 6.65

> sd(data)
[1] 6.142112

> max(data)
[1] 20

> min(data)
[1] 2

> range(data)
[1]  2 20

> quantile(data)
   0%    25%    50%    75%    100%
2.00   3.75   6.65   10.25   20.00
```

可以使用summary()函数一次性计算均值、中位数和四分位数：

```
> summary(data)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
2.000   3.750   6.650   8.162  10.250  20.000
```

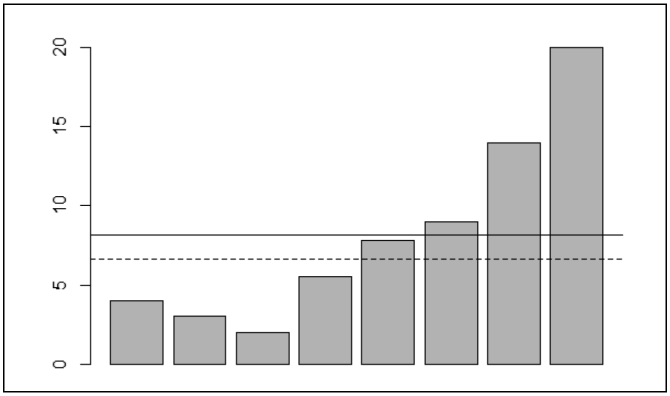
通过统计图对数据进行可视化。柱状图是一种基本的统计图，生成柱状图后，可以使用abline()函数为柱状图加入均值和中位数。因为缺省的线型是实线，所以使用参数lty=2为中位数指定线型为虚线，以区别于均值实线：

```
> barplot(data)

> abline(h = mean(data))

> abline(h = median(data), lty = 2)
```

上述命令输出如下。



有多种函数可以生成不同的数据分布。下面看一下生成正态分布的函数rnorm()，以下代码可以生成均值为0、标准差为1的100个数据点。画出这些数据点，然后做出柱状图。此外，为了能够复制结果，要确保使用同一个随机数种子，可以用set.seed()设置种子：

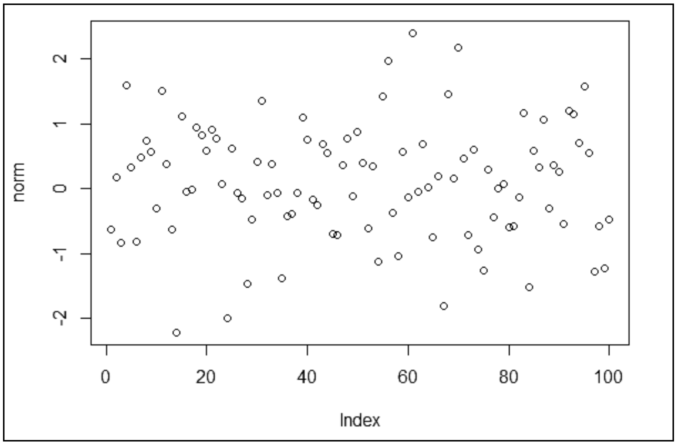
```
> set.seed(1)
```

```
> norm = rnorm(100)
```

绘制这100个数据点：

```
> plot(norm)
```

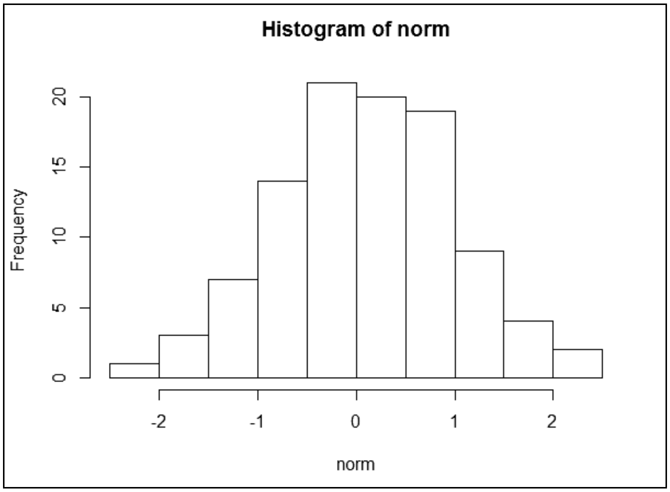
上述命令输出如下。



最后，使用hist(norm)生成柱状图：

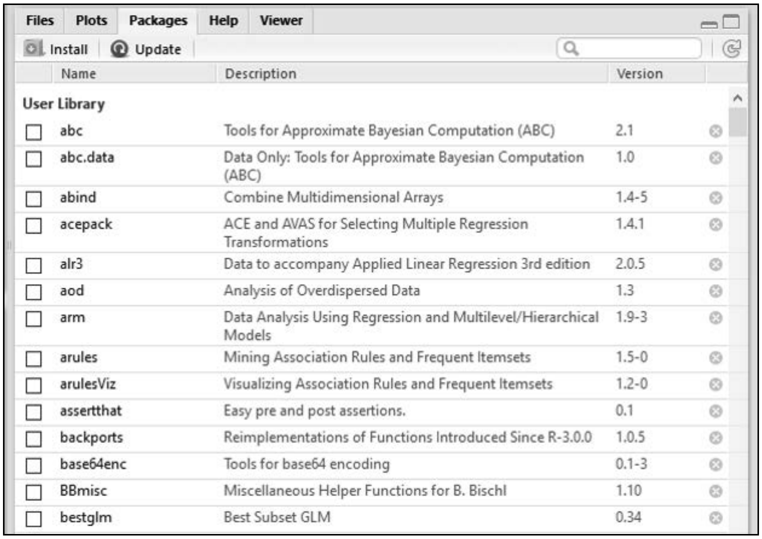
```
> hist(norm)
```

上述命令输出如下。

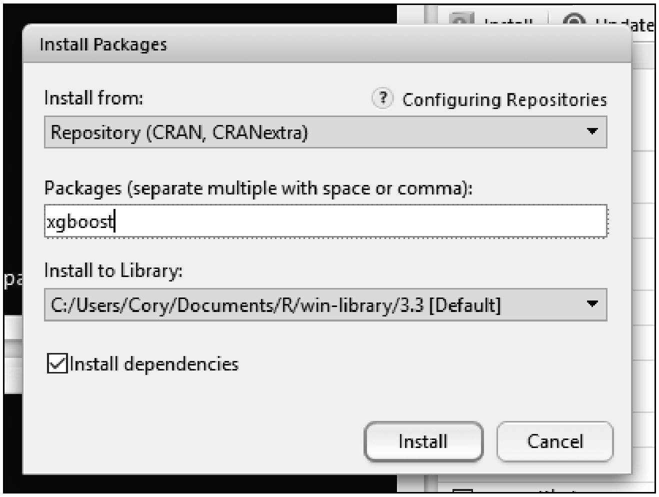


5. 安装与加载R程序包

前面已经讨论了如何使用`install()`函数安装R的程序包。要想使用已经安装好的程序包，需要先进行加载。我们再演示一下，首先使用RStudio进行安装，然后加载程序包。找到**Packages**标签页并点击，你会看到类似于下图的界面。



现在安装xgboost程序包。点击**Install**图标，在弹出窗口的**Packages**部分输入程序包名称。



点击**Install**按钮。程序包安装完成之后，弹出窗口就会返回。要想加载并使用程序包，只需使用`library()`函数：

```
> library(xgboost)
```

这样即可使用程序包中的函数。

6. 使用dplyr进行数据处理

在过去的几年里，我越来越多地使用dplyr包，对数据进行处理和总结。这个包的速度比R基础函数快，允许将多个函数连接使用。一旦熟悉了它的语法，就会越来越感受到其用户友好性。根据我的经验，仅要很少几个函数就可以完成所需的大部分数据处理工作。按照前面描述的方法安装这个软件包，然后加载到R语言环境：

```
> library(dplyr)
```

对R基础包中的鸢尾花数据集（iris）进行探索。最常用的两个函数是`summerize()`和`group_by()`。在以下代码中，你会看到如何生成一个按Species分组的包括Sepal.Length的均值的表格。均值保存在变量`average`中。

```
> summarize(group_by(iris, Species), average = mean(Sepal.Length))
  # A tibble: 3 X 2
     Species average
      <fctr>   <dbl>
1     setosa   5.006
2 versicolor   5.936
3  virginica   6.588
```

有多种摘要函数：`n`（计数）、`n_distinct`（唯一计数）、`IQR`（四分位距）、`min`（最小值）、`max`（最大值）、`mean`（均值）和`median`（中位数）。

dplyr允许使用管道操作符`%>%`帮助阅读代码。通过管道操作符，可以将函数连接在一起，从而避免函数彼此之间的包装。可以从要使用的数据框开始，将多个函数连接在一起，使得第一个函数的函数值或参数可以传递给下一个函数，以此类推。下面的例子演示了如何使用管道操作符得到和前面同样的结果：

```
> iris %>% group_by(Species) %>% summarize(average =
  mean(Sepal.Length))
  # A tibble: 3 X 2
     Species average
      <fctr>   <dbl>
1     setosa   5.006
2 versicolor   5.936
3  virginica   6.588
```

`distinct()`函数可以找出一个变量中的唯一值。看看变量`Species`中有哪些不同的值：

```
> distinct(iris, Species)
     Species
1     setosa
2 versicolor
3  virginica
```

使用`count()`函数，自动对变量进行分类计数：

```
> count(iris, Species)
  # A tibble: 3 X 2
     Species     n
      <fctr> <int>
1     setosa    50
2 versicolor    50
3  virginica    50
```

如何基于一个匹配条件选取特定行呢？可以用`filter()`函数完成这个任务。选取Sepal.Width大于3.5的行，放入新数据框：

```
> df <- filter(iris, Sepal.Width > 3.5)
```

看一下这个数据框。首先按照Petal.Length的值对行进行降序排列：

```
> df <- arrange(iris, desc(Petal.Length))

> head(df)
  Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1          7.7         2.6          6.9         2.3  virginica
2          7.7         3.8          6.7         2.2  virginica
3          7.7         2.8          6.7         2.0  virginica
4          7.6         3.0          6.6         2.1  virginica
5          7.9         3.8          6.4         2.0  virginica
6          7.3         2.9          6.3         1.8  virginica
```

下面选出感兴趣的变量，可以通过`select()`函数完成。具体要求是创建两个数据框，一个由Sepal开头的列组成，另一个由Petal开头的列和Species列组成。换句话说，由不以Se开头的列

组成。在select()函数中指定这些特定的列即可完成上述任务，也可以换种方式——使用starts_with语法：

```
> iris2 <- select(iris, starts_with("Se"))

> iris3 <- select(iris, -starts_with("Se"))
```

接着合并这两个数据框。还记得前面用过的cbind()吗？你还可以使用dplyr包中的bind_cols()函数，它能够合并两个数据框。请注意，这个函数和cbind()一样，是按位置匹配行的。如果有行名称或其他关键字，比如客户ID等，你还可以使用left_join()和innner_join()这样的函数来连接数据。因为这两个数据框中的行数是相等的，所以这个函数可以顺利运行：

```
> theIris <- bind_cols(iris2, iris3)
  head(theIris)
  head(iris)
```

可以使用head()函数比较theIris和iris两个数据框的前6行，会看到它们完全一样。如果你想像之前使用rbind()函数那样连接数据，可以使用bind_rows()函数。如果想知道Sepal.Width中有多少个唯一值，应该怎么做呢？回忆一下，数据集中一共有150个观测。我们已经使用了distinct()和count()。下面这段代码可以找出唯一值的确切数量——23：

```
> summarize(iris, n_distinct(Sepal.Width))
  n_distinct(Sepal.Width)
1                      23
```

几乎所有大数据集中都存在重复观测，复杂的连接操作也会产生重复数据。使用dplyr删除重复数据会非常简单。举例来说，假设想创建一个只由Sepal.Width中的唯一值组成的数据框，并在数据框中保留所有列，那么可以使用下面的代码：

```
> dedupe <- iris %>% distinct(Sepal.Width, .keep_all = T)

> str(dedupe)
'data.frame': 23 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 4.4 5.4 5.8 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 2.9 3.7 4 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.4 1.5 1.2 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.2 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1
 1 1 1 1 1
```

请注意，我使用管道操作符连接iris数据集和distinct()函数，并在distinct()函数中设定.keep_all = T，使得所有列都包括在新的数据框中，否则新数据框中就只有Sepal.Width一列。

如果你想在R中进行更有效率的数据整理，不妨试试dplyr。

7. 小结

这个附录的目的是使R初学者掌握R语言的一些基础知识，以便能看懂书中的代码。这部分内容包括如何安装R和RStudio，以及如何建立对象、向量和矩阵。在这之后，我们介绍了一些数学和统计函数，还介绍了如何使用RStudio安装和加载R程序包。最后介绍了dplyr的强大功能，以有效处理和总结数据。附录还介绍了生成统计图的基本方法，并给出了示范。这份附录不能使你成为R专家，但可以使你更好地理解书中的例子。

Amazon.com读者评论

“这是一本通俗易懂的机器学习和数据分析技巧参考书。书中的概念以清晰的逻辑形式展现，还给出了大量常被同类书所忽略的数据准备和商业案例。作者还提供了大量实用代码，非常适合自学。”

“本书附带大量实用代码。读者能保存书中代码并在必要时加载，也能修改代码以从中学习。说实话，我认为这本书是用R进行机器学习的不二参考。每章都涵盖一个问题，并逐步引入新技术和策略进行解决，然后通过实施以巩固所学知识。这能让读者看到相关技术适用于哪些情景及其使用方法，从而成为用R语言进行机器学习和高级统计的多面手。”

本书使用R语言讲述机器学习高级技术，带领读者深入研究统计学习理论和监督式学习，理解如何设计高效算法，学习构建推荐引擎，运用多类分类和深度学习，等等。通过探索数据挖掘、分类、聚类、回归、预测建模、异常检测等，本书帮助读者理解这些概念的工作原理和能够实现的操作。读者将循序渐进地学习神经网络等主题，探索深度学习等内容。通过不同方式使用不同的数据集，读者还可以在AWS等云平台上利用R亲手实践机器学习。

[PACKT]
PUBLISHING

图灵社区：iTuring.cn
热线：(010)51095186转600
分类建议 计算机/机器学习
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-47778-1
9 787115 477781 >
ISBN 978-7-115-47778-1
定价：69.00元

看完了

如果您对本书内容有疑问，可发邮件至 contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱: ebook@turingbook.com。

在这可以找到我们:

微博 @图灵教育 : 好书、活动每日播报
微博 @图灵社区 : 电子书和好文章的消息
微博 @图灵新知 : 图灵教育的科普小组
微信 图灵访谈 : ituring_interview，讲述码农精彩人生
微信 图灵教育 : turingbooks